ioProgrammo Anno IX - N°1 (87)

VISUAL C# EXPRESS 2005

TUTTO SUI NUOVI STRUMENTI DI MICROSOFT A BASSO COSTO

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90 | VERSIONE STANDARD RIVISTA+CD €6,90

ioPROGRAMMO

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • GENNAIO 2005 • ANNO IX, N.1 (87)

I TUOI PROGRAMMI VANNO IN TV

CREA FACILMENTE LA TUA PRIMA APPLICAZIONE PER:

- Windows Media Center
- Digitale Terrestre

MESSAGGI SEGRETI NELLE IMMAGINI

Un'applicazione .NET per proteggere le tue informazioni più private

MOBILE

Metti Flash nel cellulare

Ti sveliamo tutti i trucchi per programmare oggi la tecnologia che verrà

.NET

DOMINARE IL SISTEMA

Windows senza segreti con l'oggetto WMI

JAVA

INTELLIGENZA ARTIFICIALE

Scopriamo come realizzare software che impara da sè

A GRANDE RICHIESTA RITORNA

ioProgrammo WEB

- **PHP & SQLITE** La guida passo passo al nuovo database integrato con PHP 5
- **ASP.NET** Ottimizzazioni lato client per diminuire il carico sul server

VISUAL BASIC

RETE SOTTO CONTROLLO

Impara a gestirla usando le Network API di VB

JAVA

PHP & JAVA MESSAGING SERVICES

Un metodo alternativo per costruire intranet aziendali robuste

PATTERN BLUEPRINTS

Un modello per accedere ai database in modo sicuro da applicazioni distribuite

DATABASE

MYSQL & C#

Come utilizzare uno dei database opensource più potenti con gli strumenti di Microsoft

I CORSI PER IMPARARE VISUAL BASIC.NET JAVA • ECLIPSE

ALGORITMI: I SISTEMI CAOTICI COME SIMULAZIONE DELLA REALTÀ

Programmazione del Windows Media Center e del digitale terrestre • Tecniche steganografiche per nascondere le informazioni
EDIZIONI MASTER

Anno IX - n. 1 (87) Gennaio 2005

Anno IX - N.ro 1 (87) - Gennaio 2005 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
MarCom Director: Luca Perfetti
Marketing Manager: Antonio Meduri
Editor: Gianfranco Forlino
Coordinamento redazionale: Raffaele del Monaco
Redazione: Fabio Farnesi
Collaboratori: F. Bernabei, M. Bigatti, D. Bochicchio, L. Buono, C. De Sio Cesari, F. Grimaldi, F. C. Ferracchiati, A. Lacava, F. Lippo, A. Marroccelli, L. Mattei, S. Meschini, F. Mestrone, G. Natili, F. Paparoni, P. Perrotta, M. Poponi, F. Vaccaro.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Amministratore Unico: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgramma Basic (11 numeri) €49,90 sconto 35% sul prezzo di copertina di €75,90 ioProgrammo con Libro (11 numeri + libro) €79,90 sconto 40% sul prezzo di copertina di €133,90
Offerte valide fino al 31/01/05
ESTERO: Abbonamento Annuale: ioProgrammo Basic (11 numeri): € 151,80. ioProgrammo Plus (11 numeri + 6 libri): € 257,00
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Dicembre 2004

Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

Questo mese su ioProgrammo

▼ ioProgrammo 2005 RC1

Amici di ioProgrammo grazie! Prima di tutto grazie della fiducia che avete riposto in noi. ioProgrammo è la rivista più letta dagli sviluppatori italiani e lo è ormai da molto tempo. Qualche anno fa, in uno dei miei primi editoriali proponevo come obiettivo di ioProgrammo quello di diventare il punto di riferimento per la straordinaria community dei programmatori italiani, che allora come oggi era forte, professionale e meritevole di un prodotto curato nel dettaglio. A distanza di molti numeri, posso oggi affermare di sentire intorno ad ioProgrammo l'esistenza concreta di questa community. Si manifesta nelle centinaia di email che arrivano ogni giorno in redazione, si legge nella qualità degli interventi sul forum di ioProgrammo, è tangibile nel numero straordinario di copie vendute. Nonostante questo abbiamo ancora voglia di migliorare, di proporre contenuti sempre più interessanti, di fornire servizi sempre più dettagliati. In questo numero di ioProgrammo troverete alcune novità. In una normale evoluzione di un software questa che leggete potrebbe essere considerata una RC1, una versione in cui incominciano a intravedersi i cambiamenti che sostanzieranno la versione finale, attendiamo le vostre impressioni, le vostre richieste, per capire se la direzione intrapresa è quella che soddisfa di più le vostre esigenze. Da questo numero in poi troverete molto spesso negli articoli delle guide a passi, che vi consentono di avere un occhio di insieme su una tecnica che viene poi approfondita negli articoli. Noterete già un rinnovato legame fra il software presente nel CD e la rivista. Da non sottovalutare il ritorno di ioProgrammo WEB, in questo numero gli articoli riguardanti la programmazione Internet, occupano ancora uno spazio ridotto , ma nel corso del tempo dedicheremo loro uno spazio sempre maggiore. Non meno importante il ritorno degli stumenti di Macromedia che non sono da considerarsi solo dedicati al Web, ma come vedremo sono ormai degli ambienti completi pronti per essere usati anche nellla programmazione standalone. Infine, ultimo ma non meno importante, in questo numero presentiamo in modo molto rapido Python, segno che torneremo ad occuparci anche di linguaggi di scripting. Insomma ioProgrammo si rinnova ancora una volta, rispettando quelle che sono le tendenze di un mondo vivo e in fermento come è quello dei programmatori. In questo numero non troverete come di consueto un articolo di elettronica. Attenzione non è questa una novità. Torneremo come sempre nei numeri a seguire ad occuparci del legame che la programmazione ha con il mondo dell'elettronica. Attendiamo le vostre riflessioni, fermamente decisi ad informare, a supportare la crescita di questo popolo di programmatori che è e rimarrà sempre un vanto dell'Italia tecnologica.

Fabio Farnesi

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **iopusr821** *Password:* **sk31f48t**

I TUOI PROGRAMMI VANNO IN TV

Crea la tua prima applicazione per

- **Windows Media Center**
- **Digitale Terrestre**

pag. 14

METTI FLASH NEL CELLULARE

Ti sveliamo tutti i trucchi per iniziare a programmare ora la tecnologia che verrà pag. 26

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

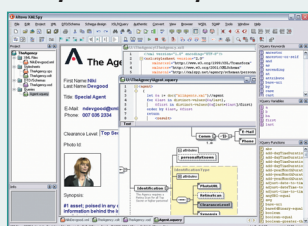
Vistita il forum di ioProgrammo all'indirizzo http://forum.ioprogrammo.it. Non perdere l'occasione di approfondire gli articoli con gli esperti presenti nel forum

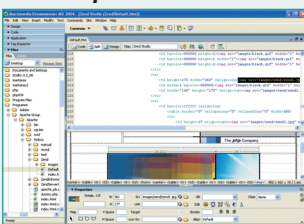
I contenuti del CD-Rom

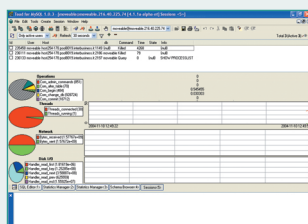
Prodotti del mese

Flash MX 2004
Multimedia senza confini

Macromedia Flash è ormai un prodotto maturo. Alle sue funzionalità classiche rivolte alla programmazione di contenuti multimediali, si sono aggiunte funzionalità relative alla gestione dei database, di XML, estensioni per lo sviluppo di applicazioni Mobile. Si tratta di un vero ambiente di supporto al potentissimo linguaggio ActionScritp.
File: FlashMX2004-x.zip [pag.26]

XMLSpy 2005
Tutto quello che serve per XML

XMLSpy si presenta come un tool entry level per lo sviluppo di documenti XML. XMLSPY E è tra i migliori tool disponibili: è possibile validare documenti XML sulla base di DTD ed è possibile trasformare i documenti utilizzando regole XSL, il tutto attraverso un apposito editor DTD, un editor grafico per XML Schema ed un processore XSLT.
File: XMLSpyent2005.exe [pag. 26]

DreamWeaver MX 2004
Uno Must per il Web

Difficile etichettare Dreamweaver con un semplice: "Editor Wysiwyg" per il Web. Dreamweaver è molto di più. È ormai uno standard de facto, utilizzato per produrre siti di media e grande dimensione, meritevole dell'attenzione che solitamente si da ad un linguaggio o ad un ambiente di programmazione.
File: Dreamweavermx-it2004.zip

Toad For Mysql
Linea di commando addio

È un tool grafico per l'amministrazione di database. Un ottimo Front-end dotato di un Sql Editor, uno schema browser, un utilissimo gestore delle sessioni e altri tool interessanti come per esempio un comparatore di Database che individua tutte le differenze fra un database e un altro.
Directory: /toad

INTERNET

Amsn 0.94
Un clone di MSN Messenger, completo di sorgenti
Directory: /amsn

Azureus 2.2.0.0
Il client BitTorrent multipiattaforma
File: Azureus_2.2.0.0_Win32.setup.exe

Drupal 4.5.0
Metti in piedi il tuo blog in 5 minuti
File: drupal-4.5.0.tar.gz

Elicpse Trader 0.90
Un plugin per eclipse per costruire applicazioni di stock trading
File: eclipsetrader-0.7.0-win32.zip

Eggdrop 1.6.17
Gestisci un canale irc con il tuo bot programmabile
File: eggdrop1.6.17.tar.gz

PHPBB 2.0.10
Un forum preconfezionato in PHP, pronto per essere usato
File: phpBB-2.0.10.zip

PHPMQ 0.23 pag. 46
Una serie di librerie per PHP che consentono di utilizzare JMS
File: phpmq_0.2.3_bin.zip

WordPress 1.2.1
Un ottimo tool per la creazione di blog
Directory: /wordpress

Deep Log Analyzer 1.51
Raccogli e analizza le statistiche sul traffico del tuo sito Web
File: Dlatrial.exe

Quick 'n Easy FTP Server 2.4.1
Un server FTP facilissimo da configurare
File: Ftpserver.exe

STRUMENTI

Bugzilla 2.18 pag. 119
Gestisci i bug scovati nei tuoi programmi
File: bugzilla-2.18rc3.tar.gz

OpenWFE 1.4.5
Un engine Workflow per java. Automatizza il processo di lavoro necessario per lo sviluppo
Directory: /openwfe

Filezilla 2.2.9
Un client FTP freeware e completo di sorgenti
Directory: /filezilla

Filezilla Server 0.9.3
Un server FTP affidabile e completo di sorgenti
File: FileZilla_Server_0_9_3.exe

Hibernate 2.1.6
Sviluppa classi persistenti in Java utilizzando un linguaggio comune
File: hibernate-2.1.6.zip

JESS pag. 80
La libreria che consente di realizzare sistemi esperti con Java
Directory: /Jess

BTR Wizard pag. 110
Sostituire una stringa in centinaia di file contemporaneamente con un clic
Directory: /Btrwizard

JasperReports 0.6.1
Una libreria scritta in java per creare report personalizzati
File: jasperreports-0.6.1-project.zip

Jaxlib 0.6.3
Una libreria che fornisce una serie di funzionalità per la gestione dell'I/O
File: jaxlib-0.6.3.zip

Mantaray 1.2.1 pag. 42
Il tool per la gestione rapida di messaggi JMS
File: mantaray_1.2.1_bin.zip

Marathon 0.90
Un ambiente di test per applicazioni scritte usando Java Swing
File: marathon-0.90-src.tar.gz

DrPython 3.6.10 pag. 117
Un editor per programmatori, scritto in Python
Directory: /drpython

PMD 2.0
Un analizzatore di software scritti in

I contenuti del libro

C++ .NET

.NET e C# rappresentano sicuramente un'evoluzione nei tool di sviluppo progettati da Microsoft. Tuttavia esiste una base di programmatori C++ piuttosto consistente e che non è per niente convinta della necessità di migrare a C#. Cosciente di questa situazione, Microsoft ha inserito nel framework .NET il supporto per C++. In sostanza è ancora possibile programmare in C++ usufruendo del supporto di un ambiente evoluto come Visual Studio.NET.
Il libro che presentiamo allegato alla versione plus di ioProgrammo è un reference rapido da consultare per avere informazioni immediate sull'uso di questa o quella caratteristica del linguaggio o dell'ambiente. Si tratta certamente di un "compatto" da posizionare di fianco al pc, pronto da consultare ogni qual volta si presenta un'incertezza durante la stesura del proprio codice.

RIVISTA + LIBRO + CD-ROM in edicola

Prova subito

XletView 0.36

L'emulatore del digitale terrestre. Per testare le tue applicazioni Java!

pag. 19

Librerie e strumenti

- ASP.NET
- VB.NET
- C#
- Flash
- Java
- PHP
- Python
- Database
- Internet
- Compilatori

pag. 110

JAVA, trova le variabili inutilizzate e molto altro
Directory: /PMD

SevenZip 4.10
Uno dei programmi con migliore algoritmo di compressione per file. Dotato ovviamente di sorgenti.
Directory: /sevenzip

Tease 1.2.1
Un editor di testo scritto da programmatori TCL per sviluppatori TCL
Directory /tease1.2.1

WinMerge 2.2
Effettua una comparazione di due file di testo anche di grandi dimensioni
Directory: /winmerge

wxAll 2.5.3 pag. 117
wxWidgets è un framework che facilita lo sviluppo di multipiattaforma
Directory: /wxall

Eyebol Delphi Analyzer 1.20
Ottimizza il tuo codice Delphi
File: Eyetry.exe

Java Launcher 1.5
Fai partire le tue applicazioni Java con un doppio-clic
File: Javalauncher.zip

MaSal Editor 1.7.1649
Un tool per creare file MSI di installazione
File: Masaieditor.exe

BlueJ 2.0.2 pag. 113
Un ambiente che introduce alla programmazione Java per via grafica
File: Bluejsetup.exe

GRAFICA

Graphics 3D++ 6.0.4
Una libreria grafica per gli sviluppatori di VideoGiochi
Directory: /G3d

Limnor 3.3
Per programmare per via grafica, senza linguaggi
File: Limnor.zip

LINGUAGGI

PHP pag. 14
Le nuove versioni di uno dei linguaggi più utilizzati nel Web
Directory: /php

Mono 1.0.4
Il clone GPL del framework .NET di Microsoft
File: mono-1.0.4-gtksharp-1.0.4-win32-0.1.exe

Python 2.3.4 pag. 117
Un linguaggio di programmazione di cui sentiremo parlare a lungo
Directory: /Python

DATABASE

HSQL Database engine 1.7.2
Un database relazionale scritto in Java
Directory: /hsqldb

Ibatis Database Layer 2.0.7
Ti aiuta ad implementare una base per la persistenza dei dati
File: iBATIS_DBL-2.0.7.459.zip

Mysql 4.1.7 pag. 48
Il famoso server di database utilizzatissimo sul Web
File: mysql-4.1.7-win.zip

SCIENTIFICI

IT++ 3.8.0
Una libreria matematica per processare i segnali
File: it++3.8.0.tar.gz

News

OTTO MILIARDI DI PAGINE INDICIZZATE

Questi sono i numeri di Google. Il fatidico traguardo campeggia in basso, sotto la casellina di ricerca del motore più usato al mondo. L'evento è enfatizzato sul Blog di Google. Ebbene sì, anche Google ha un blog, all'indirizzo: *http://www.google.com/googleblog*. Così sul Blog di Google trionfa l'annuncio del fatidico traguardo. Eppure in Google non si limitano ad aumentare le pagine indicizzate. Si lavora ormai in più campi, alcuni strumenti di ricerca sono ormai noti ai più, come la *Gmail* o la *Desktop Bar,* altri sono meno noti come ad esempio le estensioni che Google starebbe mettendo a punto per consentire agli editori di associare uno o più libri per ogni ricerca significativa. Ma, soprattutto, si continua a lavorare anche sulla qualità dell'informazione richiesta, che resta il core business dell'azienda, quello da cui derivano anche i maggiori ricavi.

DIECI NUOVI BUCHI NELL'SP2

L'accusa questa volta non è confermata: a individuare le 10 nuove "insicurezze" attribuibili al tanto acclamato SP2 sarebbero state indicate da *"Finjan Software"* a *"Microsoft"*. Tuttavia, la compagnia di Bill Gates avrebbe dichiarato di non essere al momento a conoscenza di nessun attacco basato sulle tecniche indicate da *Finijan.*
Ciononostante, alla Microsoft tengono in considerazione ogni segnalazione e sono pronti a intervenire con un opportuno Fix, se necessario.

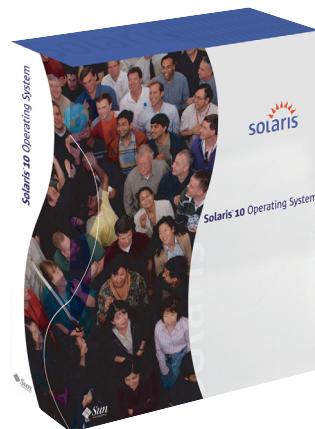
UDDI TORNA A SPLENDERE

Nato come un indispensabile sistema di pagine gialle virtuali, si riprometteva di aiutare gli sviluppatori a trovare i servizi Web nell'oceano di quelli disponibili. In tale settore, è stato decisamente trascurato. UDDI pare ora risorgere: utilizzando i Web Services è necessario che le applicazioni sappiano esattamente "da chi" i servizi siano erogati, cosa che crea un accoppiamento negativo che impone, in caso di trasloco del Web Service, l'aggiornamento di tutte le applicazioni che a quel servizio puntano. Se invece le applicazioni puntano al registro UDDI, il problema è di colpo risolto!

VENDEVA IL CODICE DI WINDOWS: ARRESTATO!

Così William O. Genovese Jr, è stato pescato con le mani nel sacco, mentre bellamente vendeva a destra e a manca porzioni di codice di Windows NT e Windows 2000. Non è ben chiaro se il codice venduto da Mr. Genovese fosse lo stesso che è uscito fuori dagli archivi di Microsoft con grande scalpore all'inizio di quest'anno. Il buon William vendeva il codice per appena 20$ da versare sul suo account su Paypal, l'acquisto poteva essere effettuato tramite il suo sito Internet. Peccato che uno degli ultimi acquirenti sia stato proprio un agente dell'FBI che stava indagando sulle fuoriuscite di codice dagli archivi di Microsoft.

SOLARIS 10 RILASCIATO

I rumors sulla data di rilascia del nuovo Solaris andavano ormai moltiplicandosi di giorno in giorno. Così è arrivato anche il giorno in cui, realmente, Solaris 10 è rilasciato! Le novità principali riguardano uno stack TCP-IP completamente rinnovato e l'adozione del nuovo *Z File System.* Ambedue le innovazioni sono tese a migliorare le performance del sistema. Il modello con cui Solaris 10 viene proposto al mercato è quello dell'Open Source. È infatti possibile scaricare e installare Solaris in modo del tutto gratuito.
A fianco alla versione base, esistono una serie di "pacchetti supporto" che hanno un costo variabile, dipendente dal livello di supporto richiesto.

NOVELL DI NUOVO CONTO MICROSOFT

Questa volta, il terreno di scontro è il word processing: WordPerfect contro Microsoft Word. Al solito, l'accusa che Novell muove contro Microsoft è quella di avere sfruttato il proprio monopolio sul mercato del software per adottare politiche lesive della concorrenza. In particolare, Novell accusa Microsoft di avere introdotto nei propri sistemi elementi integrati quali Internet Explorer ed avere agito in maniera tale che lo sviluppo di applicazioni come WordPerfect non potesse essere portato avanti con profitto. Microsoft avrebbe integrato le sue tecnologie in modo "Esclusivo" impedendo a chiunque di poter distribuire le proprie applicazioni. A riprova di questa tesi, Novell sostiene che nel 1990 WordPerfect deteneva poco meno del 50% del mercato, per scendere poi al 30% nel 1994 e a meno del 10% nel 1996 quando Novell decise di vendere WordPerfect e le applicazioni correlate. Nello stesso periodo, Microsoft Word salì da meno del 20% del 1990 a circa il 90% del 1996 in virtù delle politiche attuate da Microsoft.

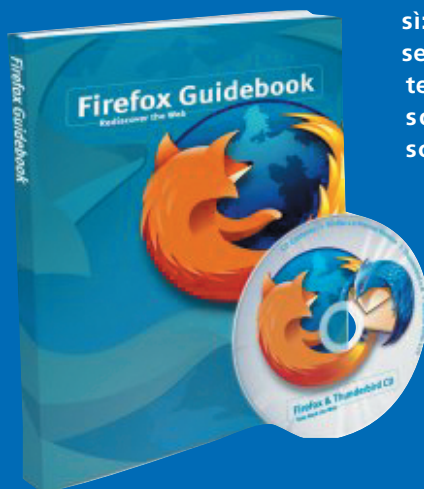
IBM AGGIORNA L'AUTONOMIC TOOLKIT

Un sistema informatico è detto "autonomo" quando riesce a riconoscere e risolvere i propri problemi, auto-riparandosi.
Senza l'intervento umano, possibilmente non interrompendo i servizi che eroga. IBM crede molto in questo approccio, soprattutto nel tentativo di dominare la crescente complessità dei dipartimenti informatici delle medie e grandi aziende. Già dall'inizio del 2004 è disponibile un apposito toolkit gratuito, integrabile in Eclipse, che aiuta gli sviluppatori a costruire sistemi autonomi.
IBM ha da poco rilasciato la versione 2 del toolkit che, tra le tante novità, migliora l'integrazione con Eclipse attraverso nuovi plug-in e aggiunge il supporto ad Eclipse 3.0. Gli altri miglioramenti hanno riguardato soprattutto la semplicità e la capacità di risalire più a fondo nell'analisi degli errori e delle cause che li hanno generati. IBM garantisce che con il nuovo toolkit che sarà più facile costruire e distribuire applicazioni con estese capacità di auto-gestione.

WINAMP ADDIO

Si chiude un'era nel mercato dei player. NullSoft, la software house produttrice del diffusissimo Winamp, sembra ormai sull'orlo del baratro. AOL aveva acquisito NullSoft nel 1999 per la modica cifra di circa 100 Milioni di dollari.
Tuttavia il team originale degli sviluppatori di Winamp non aveva gradito l'operazione. Le ultime dimissioni importanti erano state quelle di Justin Frankel, il creatore di Winamp, l'anno scorso. Ora si consumano le dimissioni degli ultimi programmatori che erano rimasti a sviluppare il prodotto. Allo stato attuale non è certo se il software continuerà ad essere sviluppato o rimarrà fermo allo stato attuale. Una nota degli sviluppatori rimasti in NullSoft sottolinea che il team è semplicemente ridotto, ma è ben lungi dall'essere in stato di chiusura.

FIREFOX BATTE SIA OPERA CHE EXPLORER

Se da un lato Microsoft si affretta a dichiarare che il suo Browser non è meno sicuro degli altri, si moltiplicano le voci di preoccupazione per il futuro di Opera. L'arrivo della versione finale di Firefox è stato un duro colpo per tutti: stabile, leggero, sufficientemente sicuro, Firefox è stato scaricato da milioni di utenti immediatamente dopo l'annuncio del suo rilascio. Contemporaneamente non sono mai cessate le critiche alla sicurezza si Internet Explorer, nonostante le barricate innalzata da Microsoft intorno al suo Browser.
Per quanto riguarda Opera, pur trattandosi di un ottimo browser, sembrerebbe in grave crisi economica, pare soprattutto a causa della fiacchezza del dollaro. In ogni caso, negli ultimi anni c'eravamo abituati a vedere Internet Explorer dominare il mercato. Non è più così: sembrano essere tornati i bei tempi in cui Netscape e Microsoft si battevano sul piano dei browser. Chi vincerà questa volta? Explorer, Opera, Firefox? O addirittura browser come Safari?

MICROSOFT ANNUNCIA LA PROPRIA DESKTOP SEARCH BAR

Ormai è guerra totale. Se da un lato Google investe sul mercato annunciando prodotti rivoluzionari come la *Desktop Search Bar*, affondando così il colpo sul mercato del software Standalone, dall'altro lato Microsoft risponde sia proponendo una versione rivista e migliorata del proprio motore di ricerca, sia contrapponendo una propria Desktop Search Bar. Nel più puro stile Redmond, lo fa acquisendo una piccola compagnia, la LookOut, che aveva già sviluppato un Tool che consentiva agli utenti di Outlook 2000+ di effettuare velocemente delle ricerche molto accurate all'interno della propria posta elettronica.
In realtà la MSN Toolbar Suite, al momento in cui scriviamo, non è stata ufficialmente annunciata da Microsoft, anche se circolano già in rete indiscrezioni sulle sue funzionalità e persino alcuni screenshot. Il rilascio ufficiale è previsto per la fine del mese di Dicembre, salvo ritardi dell'ultimo minuto.

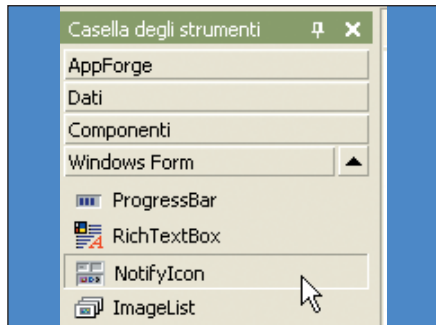
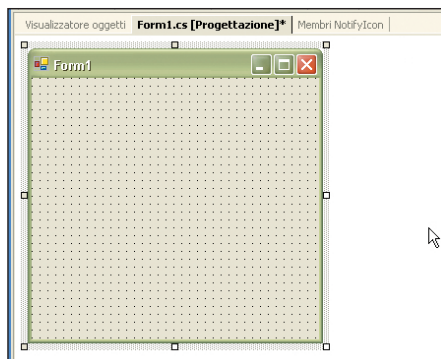
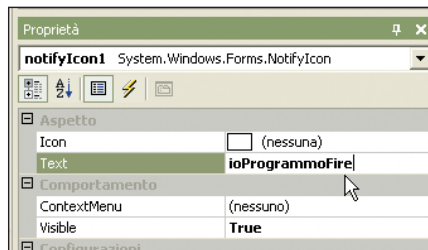
INBox L'esperto risponde...

Metti una .NET nella system tray

Cari amici di ioProgrammo, grazie ai vostri stupendi articoli ho cominciato a muovere i primi passi nel mondo della programmazione .NET. Ora sono alle prese con una piccola applicazione che gira in background e vorrei che comparisse solo una piccola icona nella barra di Windows in basso a destra, mi pare si chiami "System Tray". Come fanno molti antivirus e firewall. Mi dareste una mano?

Marco De Gennaro

Risponde la Redazione

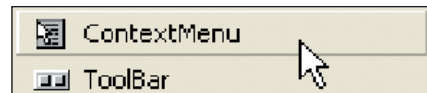
Caro Marco, con Visual Studio è particolarmente semplice realizzare quanto chiedi: apri il progetto relativo alla tua applicazione e, dalla casella degli strumenti, all'interno di Windows Form, scegli NorifyIcon e trascinalo sulla tua form.

Non essendo un controllo visibile, vedrai apparire in basso una barra con all'interno un controllo dal nome notifyIcon1.

Cambiando la proprietà Text del controllo, andrai a definire la scritta che comparirà quando l'utente passera con il mouse sull'icona.

La proprietà Icon del controllo notifyIcon1 ci consente invece di specificare quale sarà l'icona che caratterizzerà la nostra applicazione nella System Tray.

Le applicazioni che hai citato (firewall, antivirus), quando minimizzate, non appaiono nella barra dei task di Windows ma, appunto, solo nella System Tray. Per ottenere questo effetto, è necessario intercettare il momento in cui l'applicazione viene minimizzata: aggiungiamo dunque un event handler per l'evento Resize della form. L'handler si occuperà di nascondere l'applicazione nel momento in cui l'utente minimizzerà la finestra. Questo il codice:

```
private void Form1_Resize(object sender,
                          System.EventArgs e)
{   if (FormWindowState.Minimized ==
                               WindowState)
      Hide();
}
```

È giunto il momento di gestire la riattivazione dell'applicazione. Se vogliamo che ciò avvenga con un doppio click sull'icona nella system tray, dobbiamo evidentemente gestire l'evento "doppio-clic" della NotifiyIcon (NotifyIcon.DoubleClick):

```
private void notifyIcon1_DoubleClick(
         object sender, System.EventArgs e)
{   Show();
    WindowState = FormWindowState.Normal;
}
```

Non resta che aggiungere il classico menu contestuale che appare cliccando con il tasto destro sull'icona: sempre dalla casella degli strumenti di Visual Studio, puoi trascinare il controllo ContextMenu nella tua form. Accanto a notifyIcon1 dovrebbe ora apparire un contextMenu1. Un clic

con il tasto destro su quest'ultimo controllo e scegliamo la voce "ModificaMenu" per aggiungere le voci che ci interessano. Come al solito, per aggiungere il codice relativo ad ogni voce, sarà sufficiente un doppio clic sulle stesse. Infine, clicchiamo su notifyIcon1 e, tra le proprietà, andiamo a modificare ContextMenu: nel menu a discesa, selezioniamo il contextMenu1 appena creato.

Ping da Java

Gentile redazione di ioProgrammo, vorrei richiamare all'interno della mia applicazione il Ping di Windows per verificare la presenza di un PC in Rete. Come potrei fare?

Tullio Di Girolamo

Risponde la Redazione

Gentile Tullio, per richiamare il Ping di Windows, puoi utilizzare la classe Runtime che consente l'avvio di una qualsiasi applicazione. In questo caso, il valore ritornato dal metodo exec di Runtime può essere utilizzato per verificare l'avvenuta connessione al PC remoto.

```
import java.io.*;
import java.net.*;
class Ping {
  public static void main(String[] args) {
    BufferedReader in = null;
    try {
      Runtime r = Runtime.getRuntime();
      Process p = r.exec("ping 150.17.0.25");
      if (p == null) {
        System.out.println("Nessuna
                              connessione");}
      in = new BufferedReader(new
       InputStreamReader(p.getInputStream()));
      String line;
      while ((line = in.readLine()) != null) {
        System.out.println(line); }
      in.close();
    } catch (IOException io) {
      System.err.println(io.toString());}}
}
```

MySQL 4.1 non funziona con PHP 4

Buongiorno, da poco sto cercando di imparare a usare PHP e ovviamente lo uso in coppia con MySQL. Da quando ho installato MySQL4.1, le abituali modalità di connessione al database non funzionano più.
È variato qualcosa? Se utilizzo ad esempio:

```
<?php
  /* Connessione e selezione del
                          database */
  $connessione = mysql_connect(
   "localhost", "mionome", "miapassword")
     or die("Connessione non riuscita");
  print "Connesso con successo";
  mysql_select_db("mio_database") or
          die("Selezione del database non
                           riuscita");
?>
```

Ottengo

Warning: mysql_connect(): Client does not support authentication protocol requested by server; consider upgrading MySQL client in c:\programmi\apache group\apache\htdocs\mysql\index.php on line 3 Connessione non riuscita

Che vuol dire, che devo upgradare il client? quale client?

Risponde la Redazione

Grazie per il "preziosi". Abbiamo provato a riciclarci come ornamento per dolci donzelle, ma ne abbiamo ricavato solo qualche sonora pernacchia.
In ogni caso. MySQL 4.1 usa un protocollo di autenticazione che si basa su un algoritmo di hashing incompatibile con i vecchi client. PHP nelle versioni fino alla 4.3.9 utilizza la vecchia modalità di gestione dell'autenticazione, meno sicura. Al momento le soluzioni sono le seguenti:

1) Upgradare ad una nuova versione del client che sia in grado di gestire i metodi di autenticazione di MySQL 4.1

oppure

2) Riportare le password degli utenti del database alla vecchia modalità, tramite la funzione *OLD_PASSWORD*

```
c:\mysql -uroot -plapasswordcorretta
mysql> SET PASSWORD FOR
    -> 'some_user'@'some_host' =
              OLD_PASSWORD('newpwd');
```

È possibile anche modificare tutte le password riportandole al vecchio stile in un'unica soluzione

```
mysql> UPDATE mysql.user SET Password =
               OLD_PASSWORD('newpwd')
  -> WHERE Host = '*' AND User = '*';
mysql> FLUSH PRIVILEGES;
```

oppure cambiare le modalità delle password solo di alcuni utenti, utilizzando al posto dell'asterisco l'host e l'utente corretto.

Non funziona MySQL con PHP5

Gentile redazione di ioProgrammo, ho da poco installato PHP5 con MySQL 4.1. Ero abituato a che PHP e MySQL lavorassero insieme senza alcun tipo di intervento. Invece adesso quando tento una connessione ottengo:

Fatal error: Call to undefined function: mysql_connect()

Che vuol dire? MySQL non è più supportato da PHP5?
Mi pare assurdo visto che gran parte dei siti Web funzionano grazie all'accoppiata MySQL e PHP.

Risponde la Redazione

Caro lettore, PHP5 continua a supportare tranquillamente MySQL. Nelle versioni della serie 4 il supporto a MySQL era inserito in modo embedded all'interno delle versioni precompilate per Windows di PHP. Dalla versione 5 in poi invece, il supporto a MySQL non è più implicitamente compilato nei binari, ma deve essere attivato come estensione .
Questo può essere fatto eliminando il punto e virgola dalla riga:

```
;extension=php_mysql.dll
```

nel file *php.ini* contenuto nella cartella di sistema *c:\windows.*
Inoltre è importante settare il PATH di sistema per la directory che contiene *libmysql.dll.*
Il path può essere settato dal pannello di controllo, accedendo a *sistema->avanzate->variabili d'ambiente.*

I dubbi di soluzioni

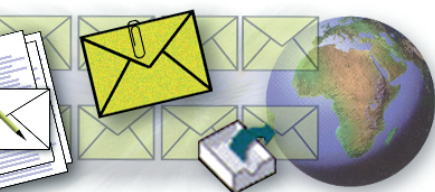
Ho affrontato il problema del giro del cavallo proposto in alcuni numeri passati di ioProgrammo. Ho implementato con successo un algoritmo che risolve il problema sulla scacchiera 8X8 dove il cavallo ha il passo classico 2,1. Tramite ricerche ho scoperto che sulla scacchiera 8X8 il cavallo dovrebbe concludere il tragitto anche con passo 2,3. Purtroppo il mio algoritmo non riesce nell'impresa. Ho provato a usare anche la formula backtracking pura proposta da steven nella sezione del forum, ma il programma sostanzialmente non termina... avete "lumi" a riguardo?

Andrea

Risponde Fabio Grimaldi

Sono molto contento che la rubrica enigma stia suscitando un vivace interesse, e il tuo contributo è un'ulteriore conferma. Nello specifico, per il problema del giro di cavallo, non sono in grado di dare una risposta esaustiva, non avendo sotto mano il codice da te prodotto. Posso però fare alcune considerazioni in base a ciò che penso di aver capito e per la mia sensibilità verso il backtracking. Innanzitutto, hai tentato una variante per la quale il cavallo si muove con il tipico movimento a *L* ma con passo più lungo, tale movimento rende più impacciato il movimento dell'equino, fare una similitudine con il caso reale non è una semplice battuta di spirito, ma realmente si colloca nella difficoltà tecnica che si incontra nella risoluzione dell'algoritmo. Anche se non supportato da studi approfonditi sulla variante da te proposta, "a naso", penso che dovrebbe garantire una soluzione per cui, se il tuo algoritmo implementa un backtracking puro, si dovrebbe pervenire al risultato. Quindi, in definitiva, consiglio di fare attenzione nell'ordine alle seguenti due attività: controllo del codice per verificare un'effettiva implementazione del backtracking; fare eseguire il codice per un tempo sufficiente (può essere anche molto lungo, si può provare a lasciare il programma in esecuzione l'intera nottata, a tale proposito ricordo che all'algoritmo è associata una complessità esponenziale).

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Ariberto, 24 - 20123 Milano

NEWSGROUP
Le informazioni nella Rete

ioProgrammo seleziona per voi le discussioni più interessanti dai newsgroup tecnici di programmazione

Java
it.comp.java

Dedicare più memoria a Tomcat

Quando faccio partire Tomcat (su Linux con JDK1.4.2), come faccio a "dedicargli" piu' memoria? Mi pare ci fosse un'opzione di Java, ma non ricordo quale...

Pietro

Risponde Lucio Benfante

È sufficiente definire la variable d'ambiente *CATALINA_OPTS* valorizzandola con le opzioni che devono essere passate alla JVM che eseguirà Tomcat.
Ad esempio, con le JVM di Sun, per definire la dimensione massima dell'heap a 256 MB:

```
export CATALINA_OPTS=-Xmx256m
```

La descrizione delle altre opzioni non standard disponibili la trovi nella documentazione o eseguendo il comando:

```
java –X
```

Il JPanel si sposta

Sapreste spiegarmi perché quando cambio il contentPane di un JFrame con menu, il JPanel chiamato va a finire per il pezzo iniziale nascosto sotto il menù?

Daniele 80

Risponde Marco

Creo la gui in un certo momento al click di un pulsante sul frame e faccio:

```
setContentPane(new MiaClasse());
```

Spegnere Windows XP

Qualcuno è in grado di suggerirmi del codice o documentazione su come creare un metodo capace di spegnere e riavviare WinXP da un'applicazione Java?

Louis Cyphre

Risponde Vincent Vega

Prova con questa istruzione:

```
Runtime.getRuntime().exec("shutdown -s");
```

Altrimenti fai una libreria dinamica (DLL) che espone il metodo JNI compatibile e lo richiami tramite JNI.

PHP
comp.lang.php

Come creare uno script basato su timer/counter?

Buongiorno, quello che vorrei è uno script che ogni 5 minuti esegue una ricerca nel mio database e mostra una piccola finestra popup se certe condizioni sono verificate.

Massimiliano

Risponde Marco R.

È sufficiente inserire nella pagina un tag meta refresh come segue:

```
<META HTTP-EQUIV=Refresh CONTENT=
        "10; URL=http://www.miosito.it/">
```

questo effettuerà un refresh della pagina ogni 10 secondi. Perciò lo script che esegue le operazioni di confronto con il database ed eventualmente apre una finestra popup può essere rilanciato.

Come scompattare un file usando PHP?

Buongiorno a tutti, mi piacerebbe sapere come fare per decompattare un file zip direttamente da uno script php?

Luigi

Risponde eliod65

Ti consiglio due librerie interessanti, la prima *http://php.kuchingfest.com/zip/zip.phpc*, la seconda *http://www.phpconcept.net/pclzip/index.en.php*.

Visual Basic
it.comp.lang.visual-basic

Visualizzare il valore di una variabile

Come posso visualizzare il valore di una variabile in una textBox?

Marci

Risponde Marco R.

Con le convenzioni del pannello di controllo:

```
textBox.Text = Variabile
```

oppure col punto:

```
textBox.Text = ltrim$(str$(Variabile))
```

oppure per fissare i decimali

```
format()
```

Intercettare i tasti funzione

Come utilizzare i tasti funzione? Nel senso che vorrei lanciare una routine alla pressione di un tasto funzione (es. F8) ma non riesco a catturarlo. Con gli altri tasti non ho problemi, usando il codice ASCI.

Roger

Risponde Megasoft 78

Sull'evento KeyDown fai qualcosa del genere:

```
Select Case KeyCode
    Case vbKeyF1    'fai qualcosa
        KeyCode = 0
    Case vbKeyF2    'fai qualcosa
        KeyCode = 0
End Select
```

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: ioProgrammo - Edizioni Master, Via Ariberto, 24 - 20123 Milano

Una semplice applicazione in Java per decoder digitali terrestri

Televisione Digitale Terrestre Interattiva

In questo articolo, impareremo qualcosa sul digitale terrestre, parleremo delle estensioni Java per la TV e scriveremo una prima applicazione

CD WEB
DgT.zip
cdrom.ioprogrammo.it

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
Basi di J2SE e del package AWT

Software
J2SE 1.4.1 SDK o superiore, J2ME Wireless Toolkit 2.0 o superiore

Impegno

Tempo di realizzazione

L'avvento delle trasmissioni televisive in digitale viene da molti considerato come una rivoluzione tecnologica che cambierà il modo di utilizzare la televisione, grazie all'interattività e alla connessione ad internet. Vediamo come ciò sia possibile e soprattutto quali siano le capacità dei nuovi decoder.

DIVERSI STANDARD

Bisogna comprendere che il Decoder per il digitale terrestre è solo uno dei tanti apparecchi tecnologici che popolano le nostre case. Molti di questi apparecchi hanno bisogno di comunicare fra loro. Inoltre il profilo che si sta delineando per i prossimi anni è quello dell'integrazione stretta fra tutti gli apparecchi d'uso comune. Per ottenere lo scopo doveva necessariamente nascere uno standard che definisse alcuni parametri minimi che garantissero l'integrazione fra le diverse periferiche. Per quanto riguarda, decoder e televisori, lo standard che si sta affermando è l'MHP Multimedia Home Platform.
Si tratta di un insieme di specifiche che definiscono le interfacce fra le applicazioni e le periferiche su cui possono essere eseguite. Pertanto la programmazione di decoder, PC Multimediali, Set Top Box etc. si rifà agli standard dettati da MPH. MPH utilizza come base la piattaforma J2ME di Sun e per quello che concerne questo articolo le JavaTV API.
I package inclusi in questo set sono già parecchi e includono, fra gli altri:

- **javax.tv.xlet:** La classe base che gestisce il ciclo di vita dell'applicazione.
  È l'equivalente di una servlet in ambito server o di una midlet per cellulari.
- **javax.tv.graphics:** una versione quasi integrale di AWT, per gestire l'intefaccia grafica.
- **javax.tv.media:** Aggiunge le funzionalità di gestione degli stream audio-visivi. Fà parte del framework JMF che molti di voi già conosceranno.
- **javax.tv.service:** Definisce i concetti ad alto livello per descrivere ed interrogare i "Servizi" offerti dai Broadcaster. Si può accedere anche a titoli e descrizioni degli spettacoli in onda.

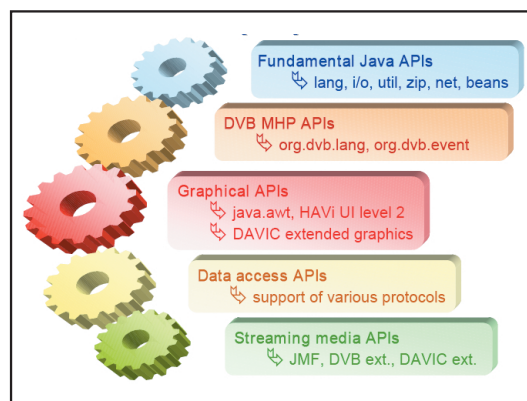



Fig. 1: *Alcuni package da utilizzare nella scrittura di una Xlet*

A queste, si aggiungono molte altre classi offerte da MHP che vedremo nello sviluppo della nostra applicazione. La maggior parte dei decoder per il digitale terrestre attualmente in commercio supporta le API MHP nella versione 1.0. Nella prossima versione, la 1.1, sarà incluso anche un browser ed un client mail con le relative classi di parsing HTML e accesso a mailboxes.

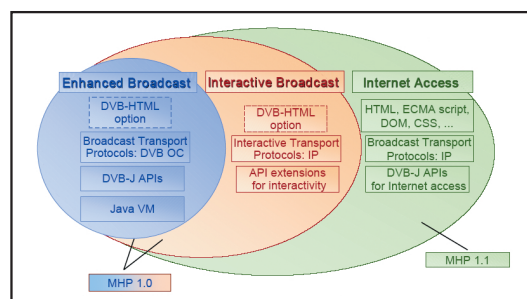



Fig. 2: *Uno sguardo ai diversi profili di classi che ci sono offerti per sviluppare una applicazione interattiva*

LA NOSTRA APPLICAZIONE

Al di la del funzionamento in termini elettronici e fisici del decoder per il digitale terrestre, scenderemo nel dettaglio di come realizzare un'applicazione che simuli il vecchio "Teletext". Immaginate che nel flusso di dati che viene convogliato verso il decoder, esista anche un canale che contenga delle notizie in formato testuale. Quello che faremo sarà semplicemente prendere questi dati dal flusso e mostrarli a video.

La nostra applicazione, o Xlet, per cominciare ad utilizzare la nuova terminologia, girerà in una SandBox, con dei vincoli di sicurezza e con i conseguenti limiti nell'accedere alle risorse di sistema. Questa impostazione è quantomeno appropriata dato che tutti siamo disposti a sopportare qualche blocco del sistema su un computer, al limite anche su un cellulare, ma sicuramente non vogliamo mettere a repentaglio il nostro sistema nervoso, perdendo il rigore della finale di Champions League per colpa di un bug di qualche applicazione! Fortunatamente, nello scrivere questo primo progetto, non verremo ostacolati da nessuno di questi limiti, dato la sua semplicità e le poche risorse a cui faremo accesso. Una Xlet, per potersi definire tale, deve implementare quattro metodi:

InitXlet	Chiamata quando il gestore delle applicazioni inizializza la Xlet.
startXlet	Chiamata quando il gestore delle applicazioni esegue la Xlet.
pauseXlet	Chiamata quando il gestore delle applicazioni decide di mettre in pausa la Xlet.
destroyXlet	Chiamata quando il gestore delle applicazioni distrugge la Xlet.

Nella *InitXlet* arà bene quindi inizializzare eventuali oggetti globali.

```
public void initXlet(XletContext xletContext) throws
                              XletStateChangeException {
  /*ci limitiamo a scrivere un messaggio di "Log" e a
                              memorizzare un riferimento
  * del contesto              */
  System.out.println("begin initXlet");
  context = xletContext;
}
```

Le operazioni vere e proprie di inizializzazione dell'applicazione preferiamo farle in *startXlet* per non occupare inutilmente risorse di sistema:

```
public void startXlet() throws XletStateChangeException {
  System.out.println("begin startXlet");
  repository = new UserEventRepository(
                                      "UserRepository");
  repository.addAllColourKeys();
  manager = EventManager.getInstance();
  manager.addUserEventListener(this, repository);
  TRANSLUCID = false;
  statoVisualizzazione = INVISIBILE;
  mostraPannello();
}
```

Avrete notato l'inizializzazione a *"INVISIBILE"* della variabile di stato di visualizzazione del pannello. Il metodo *mostraPannello()*, infatti, a seconda del valore della variabile *statoVisualizzazione* si occuperà di visualizzare o meno l'interfaccia grafica della nostra applicazione. I possibili stati in cui si può trovare la nostra applicazione sono tre:

- **Invisibile** - A schermo è presente solo una piccola immagine con un menù grafico.
- **Anteprima** - A schermo abbiamo, oltre il menù grafico, il titolo della notizia disponibile.
- **Visibile** - Un pannello che occupa gran parte dello schermo mostra la notizia.

Fig. 3: ***Il ciclo di vita di una Xlet. Ogni volta che si passa da uno stato ad un altro viene eseguito il metodo corrispondente***

Non ci soffermeremo sull'implementazione di *pauseXlet* che in progetto come questo non occorre utilizzare, a differenza di *destroyXlet* in cui, obbligatoriamente, dobbiamo liberare le risorse utilizzate e notificare la chiusura:

```
public void destroyXlet(boolean flag) throws
                              XletStateChangeException {
  System.out.println("destroyXlet");
  if (scene != null) {
    scene.setVisible(false);
    scene.removeAll();
    scene = null; }
  context.notifyDestroyed();}
```

Abbiamo già avuto modo di incontrare scene, instanza di *HScene* che riveste un ruolo fondamentale nella visualizzazione dell'interfaccia grafica, come vedremo proseguendo la codifica del nostro progetto. Attraverso il metodo *removeAll()* ci preoccupiamo di eliminare tutti gli elementi grafici dallo schermo.

SANDBOX

Letteralmente "area protetta" una zona isolata dalle altre applicazioni, tale che qualunque problema si possa verificare non influenza il resto del sistema.

Bisogna ricordare che lo schermo televisivo è ben diverso da un monitor di un pc. Anche i televisori più generosi in risoluzione, infatti, visualizzano le immagini con il metodo dell'interlacciamento. Ad ogni ciclo di refresh non vengono aggiornate tutte le linee di pixel, ma solo una parte. Per questo bisogna evitare di visualizzare linee più strette di 4 pixel e piccoli testi.

PRELEVARE I DATI

Sicuramente uno dei maggiori vantaggi offerti da questa nuova tecnologia è l'enorme banda in download. Abbiamo a disposizione, infatti, fino a 24Mbit /sec, dei quali mediamente solo 4 sono utilizzati da un singolo canale in codifica MPEG2. Oltre alla possibilità di gestire più canali video contemporaneamente, abbiamo anche i mezzi per poter accedere a grandi quantitativi di dati remoti in maniera abbastanza veloce. A questo canale di Broadcast possiamo affiancare, nei decoder più recenti, un "canale di upload", per poter permettere all'utente un'esperienza realmente interattiva. Solitamente il canale di upload si appoggia ad un collegamento ad internet attraverso un modem V90 interno. Nell'ottica di maggiore interattività e data la sempre maggiore espansione della banda larga in Italia come nel resto del mondo, un collegamento always-on via ethernet su ADSL non è da un'ipotesi troppo futuristica e sicuramente renderebbe realizzabili scenari veramente interessanti. In questo esempio, non ci preoccupiamo di utilizzare il canale di upload, limitandoci ad accedere a due files remoti, sul canale dati in broadcast. Per farlo, instanziamo un *DSMCCObject* del package *org.dvb.dsmcc* che fa parte del contributo del *DVB Project* alla causa della televisione digitale. Purtroppo nell'emulatore da noi utilizzato, *XleTView*, progetto opensource molto ben scritto, al quale hanno collaborato anche alcuni italiani, non implementa questo costrutto e consiglia, per adesso, di avvalerci delle API *java.io.File* per i nostri test. Quindi per accedere al Filesystem remoto, non utilizzeremo questo costrutto:

```
DSMCCObject carouselFile = new DSMCCObject(
                                        "dvb://notizia");
carouselFile.asynchronousLoad(this);
FileInputStream inputStream;
inputStream = new FileInputStream(carouselFile);
```

Ma ci limiteremo ad aprire il file in locale sul nostro hard disk:

```
String s = "not read";
try {
  File carouselFile = new File("notizia");
  FileInputStream inFile = null;
  inFile = new FileInputStream(carouselFile);
  StringBuffer sb = new StringBuffer();
  int b = 0;
  while( (b = inFile.read()) != -1){
    sb.append((char)b);}
  s = sb.toString();
} catch (Exception e) {}
```

In questo modo, possiamo sincronizzare il titolo e la notizia da visualizzare nel nostro televideo di "nuova generazione".

INTERATTIVITÀ

La nostra applicazione è disegnata per funzionare in un decoder interattivo. Vediamo come si può offrire interattività ad una Xlet. L'utente ha come unica preriferica di input il telecomando. Oltre ai tasti convenzionali con i numeri o per la gestione del volume, solitamente troviamo dei pulsanti funzione, diversi da decoder a decoder, e 4 tasti colorati (rosso, blu, giallo, verde), presenti in tutti i decoder indipendentemente dal produttore e dal modello. Solitamente la loro funzione non è ben chiara, ma noi possiamo sicuramente utilizzarli in qualche modo!

Abbiamo già detto di tre diversi stati in cui si può trovare la nostra applicazione: *invisibile*, *anteprima visibile* e *pannello visibile*. Vogliamo quindi gestire l'accesso ad ognuno di questi stati attraverso la pressione di un tasto funzione. In particolare, useremo il tasto giallo per vedere l'anteprima della notizia, il rosso per visualizzarla completamente e il verde per rendere invisibile l'applicazione. Al blu, dedicheremo una "finezza stilistica": visualizzare la notizia su un pannello translucido. La gestione dell'input, come spesso accade in Java, segue il paradigma ad eventi. Dichiariamo la nostra Xlet implementando l'interfaccia *listener* che controlla gli eventi "utente":

```
public class SimpleXlet
  implements Xlet, AsynchronousLoadingEventListener,
                                    UserEventListener {
  //AsynchronousLoadingEventListener serve per la
  //gestione del filesystem remoto. Inutile, dato che
  //l'emulatore non supporta quel tipo di accesso.
```

L'interfaccia *UserEventListener* è tale che a ogni evento di sistema debba essere associato il metodo corrispondente. Possiamo riconoscere quale tasto è stato premuto attraverso un suo codice ottenuto dal metodo *event.getCode()*. A seconda del tasto premuto, setteremo le variabili di stato e lanceremo il metodo di visualizzazione *mostraPannello()*:

```
public void userEventReceived(UserEvent event) {
  switch (event.getCode()){
    case HRcEvent.VK_COLORED_KEY_0:
      //ROSSO
      System.out.println("Premuto rosso");
      TRANSLUCID=false;
      statoVisualizzazione=PANNELLOVISIBILE;
      mostraPannello();
    break;
    case HRcEvent.VK_COLORED_KEY_1:
      //VERDE
      System.out.println("Premuto verde");
      statoVisualizzazione=INVISIBILE;
      mostraPannello();
    break;
    case HRcEvent.VK_COLORED_KEY_2:
      //GIALLO
```

EMULATORI

Oltre a xleTView, che trovate su: xletview.sourceforge.net potete provare DVB-MHP-reference implementato a scopo didattico da www.espial.com (al momento in cui scrivo, il sito è in rifacimento) e mhp4free che trovate su: www.mhp4free.de (sito in tedesco)

Per la messa in onda delle proprie applicazioni, bisogna ricorrere a server dedicati, i cui costi sono, ovviamente, proibitivi. Comunque, potete dare un'occhiata a: wwww.digisoft.tv che ha anche una sezione in italiano.

```
        System.out.println("Premuto giallo");
        statoVisualizzazione=ANTEPRIMAVISIBILE;
        mostraPannello();
    break;
    case HRcEvent.VK_COLORED_KEY_3:
        //BLU
        System.out.println("Premuto blu");
        statoVisualizzazione=PANNELLOVISIBILE;
        TRANSLUCID=true;
        mostraPannello();
    break; }
}
```

Ovviamente a livello di classe abbiamo dichiarato:

```
private int statoVisualizzazione;
private final int INVISIBILE =0;
private final int PANNELLOVISIBILE =1;
private final int ANTEPRIMAVISIBILE =2;
private boolean TRANSLUCID=false;
```

E ci siamo preoccupati nello startApp di chiedere alla Java Virtual Machine di catturare e notificarci la pressione dei tasti colorati:

```
repository = new UserEventRepository("UserRepository");
repository.addAllColourKeys();
```

Sebbene sia possibile catturare la pressione di qualsiasi tasto, questa operazione è poco conveniente: si impedirebbe l'accesso alle altre funzioni, persino la possibilità di cambiare canale.. quanti innocenti telecomandi rischierebbero di essere distrutti da "utenti-utonti" troppo scontrosi e poco attenti!

L'INTERFACCIA GRAFICA

In un'applicazione eseguita su una televisione un'interfaccia grafica ben curata è d'obbligo. Per questo, nonostante le risorse di un decoder siano piuttosto limitate, si è scelto di permettere la costruzione di interfacce ben più complesse di quelle realizzabili solitamente su piattaforme simili, come accade in J2ME su cellulari e palmari, per esempio. Qui possiamo avvalerci del package AWT, che tuttora è la base grafica per le applet e lo è stata per le applicazioni desktop, fino all'avvento di Swing. Il DVB project ha poi esteso queste api con il package *org.dvb.ui* che contiene diverse classi di utilità come DvbColor, per gestire colori e trasparenze. Bisogna però considerare che AWT è stato scritto per realizzare applicazioni per pc, i suoi oggetti grafici quindi sono pensati per un display VGA. Un monitor televisivo invece ha caratteristiche ben diverse, a partire dalla forma dei pixel, che non sono rettangolari. Ci viene incontro HAVI *(Home Audio Video Interoperability)*, un progetto che mira, come Jini, a coordinare l'interazione fra diversi dispositivi domestici. Le sue classi grafiche, nel package *org.havi.ui*, estendono quasi tutti i componenti AWT con versioni più adatte agli schermi televisivi. È consigliabile, quindi, ricorrere ad un HComponent invece di utilizzare un *org.awt.Component*. Le Gui Api di Havi sono incluse nelle specifiche di MHP 1.0m ma possiamo trovarle anche in vari altri dispositivi per l'home video, a partire dai decoder satellitari digitali. MHP prevede la presenza di 3 livelli grafici, sovrapposti. Il primo di questi, il Background Layer, contiene solitamente un solid color o un'immagine ferma. Ad esso si sovrappone il Video Layer, su cui vengono visualizzate le immagini in movimento del flusso video. Noi gestiremo, con le api havi, il Graphics Layer, il livello più alto. Il container in cui inserire i vari campi grafici, è costituito da *HScene*, che otteniamo da *HSceneFactory*:

```
HSceneFactory hsceneFactory =
                    HSceneFactory.getInstance();
HScene scene = hsceneFactory.getFullScreenScene(
                    HScreen.getDefaultHScreen()
                    .getDefaultHGraphicsDevice());
```

La HScene è l'equivalente di un java.awt.Frame e su di esso possiamo eseguire le operazioni di add dei vari componenti grafici: HIcon per le immagini, HText per i testi. Vediamo come gestiamo i 4 stati di visualizzazione:

```
scene.setSize(720, 576);
scene.setLayout(null);
switch (statoVisualizzazione){
  case PANNELLOVISIBILE:
    Toolkit toolkit = Toolkit.getDefaultToolkit();
    Image img = toolkit.getImage("chiudi.PNG");
    HIcon ic= new HIcon(img,590,35,96,43);
    scene.add(ic);
    HDefaultTextLayoutManager manage = new
                    HDefaultTextLayoutManager();
    HText titolo =new HText(tit, 190,18,520,20,new
        Font("Tiresias", Font.BOLD, 18),Color.blue,
                    Color.yellow, manage);
    HText text = new HText(s, 190, 40, 520, 500, new
        Font("Tiresias", Font.BOLD, 12), Color.blue,
                    colore, manage);
    text.setVisible(true);
    scene.add(titolo);
    scene.add(text);
  break;
  case INVISIBILE:
    scene.removeAll();
    toolkit = Toolkit.getDefaultToolkit();
    img = toolkit.getImage("scelta.PNG");
    ic= new HIcon(img,580,15,96,43);
    scene.add(ic);
  break;
```

Per informazioni sul mondo del Digitale terrestre:

www.mhp.org
il sito ufficiale del progetto

www.havi.org
Home Audio Video Interoperability project

java.sun.com/products/javatv/
Le specifiche di JavaTV

www.interactivetvweb.org
Qui trovate ottimi tutorial

www.dase.nist.gov/docs/javadoc/api/
alcuni javadoc sulle Havi api.

```
  case ANTEPRIMAVISIBILE:
     scene.removeAll();
     toolkit = Toolkit.getDefaultToolkit();
     img = toolkit.getImage("inantep.PNG");
     ic= new HIcon(img,580,15,96,43);
     manage = new HDefaultTextLayoutManager();
     HText titoloAnteprima =new HText(tit,
   30,520,520,20,new Font("Tiresias", Font.BOLD, 12),
        Color.blue, new Color(192,192,250), manage);
     scene.add(titoloAnteprima);
     scene.add(ic);
  break;
}
```

Importante settare "visibile" la nostra HScene:

```
scene.setVisible(true);
scene.requestFocus();
```

A seconda del diverso valore della variabile *statoVisualizzazione* avremo quindi a schermo una PNG che illustra la funzione dei tasti colorati e, se è previsto dallo stato associato, un pannello con la notizia o solo l'anteprima della stessa. Bisogna specificare le coordinate e la dimensione, in pixel, di tutti i componenti a display, ricordando che spesso, ma non sempre, la dimensione dello schermo è di 720 x 576. Per i font, le specifiche prevedono la presenza obbligata di almeno un tipo, Tiresias, e almeno quattro formati: 24, 26, 31 e 36 punti. Non siamo sicuri quindi che la nostra applicazione sarà compatibile con i decoder in commercio, dato che abbiamo usato un font a 12 punti. Comunque l'emulatore non ci darà problemi! Ultima dovuta precisazione per la compatibilità. Le specifiche MHP supportano quattro formati di immagini: png, jpg, gif e mpg (I-Frame).

RAFFINAMENTI GRAFICI: TRASPARENZE

Abbiamo già accennato alla possibilità di rendere trasparente un pannello. Questa è una funzionalità particolarmente richiesta per un'applicazione che scrive sul Graphics Layer. Possiamo, infatti, alla semplice pressione di un tasto, rendere translucido il pannello su cui è visualizzata la notizia, per non impedire completamente la visualizzazione delle immagini sul Video Layer. Per farlo, definiamo il colore di sfondo attraverso la classe *javax.tv.graphics.AlphaColor* che nel costruttore richiede, oltre il valore del colore in classico formato RGB, anche il livello di trasparenza, espresso da 0 a 255. Bisogna sottolineare che, da specifiche, non sono per forza visualizzabili tutti i valori, ma solo almeno 3 livelli: 100% (opaco), 70% (semi-trasparente), 0% (completamente trasparente). Vediamo come fare per rendere trasparente il pannello. Alla pressione del tasto blu, abbiamo settato a true la variabile booleana TRANSLUCID. Settiamo quindi il valore di sfondo del colore dell' HText text in base a questo:

```
AlphaColor colore;
if (TRANSLUCID)
   colore = new AlphaColor(192,192,255,180);
else
   colore = new AlphaColor(192,192,255,255);
```

Il colore scelto sarà comunque un azzurro (*#C0C0FF* in formato esadecimale) ma più o meno trasparente a seconda del quarto valore (*Alpha value*).

PROVIAMO L'APPLICAZIONE

Come ogni nostra creatura, anche questa merita di essere ammirata in azione, ovviamente in maniera sicura (non vogliamo bloccare nessun decoder) e soprattutto rapidamente. Non esistono moltissimi simulatori per la piattaforma MHP, che in effetti è abbastanza giovane. Uno di questi, *xleTView,* da noi scelto per questo progetto e che trovate nel CD allegato alla rivista, è un progetto opensource molto attivo e interessante. Fra l'altro ha una feature molto utile quando si fanno frequenti modifiche ad una Xlet, per esempio per effettuare piccoli cambiamenti all'interfaccia grafica. È possibile ricaricare istantaneamente la Xlet in emulazione dal menù o, ancora meglio, con la shortcut *CRTL+R*. Le librerie supportate sono parecchie, se si tiene conto di qualche accorgimento in fase di scrittura del codice. In particolare, come abbiamo già visto, l'emulatore non supporta il reperimento simulato di risorse in broadcast. Per questo abbiamo aperto il file della nostra notizia con le normali api *java.io.File*. Altro accorgimento nell'importare le classi del package *javax.tv:* il *jar* dell'emulatore contiene al loro posto le stesse classi contenute nel package *xjavax.tv*. Meglio utilizzare queste, come abbiamo fatto nella nostra semplice Xlet:

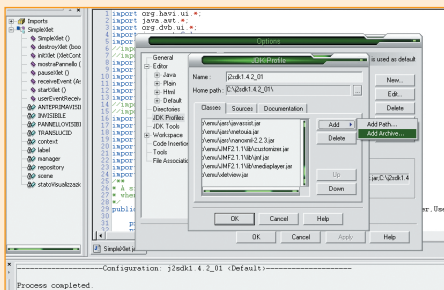
```
import org.havi.ui.*;
import java.awt.*;
import org.dvb.ui.*;
import java.awt.Color;
import java.awt.Font;
import org.havi.ui.event.HRcEvent;
import xjavax.tv.xlet.Xlet;
import xjavax.tv.graphics.AlphaColor;
import xjavax.tv.xlet.XletContext;
import xjavax.tv.xlet.XletStateChangeException;
import xjavax.tv.graphics.*;
import org.havi.ui.HDefaultTextLayoutManager;
import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
```

L'AUTORE

Luca Mattei è laureando in ingegneria informatica, appassionato del mondo java e dello sviluppo in generale, lavora come progettista software per una Mobile Company che offre la sua consulenza ai gestori di telefonia e alle major del settore ICT. È uno degli amministratori di www.JavaStaff.com, portale dedicato al mondo Java. Potete contattarlo scrivendo a: luca.mattei@javastaff.com

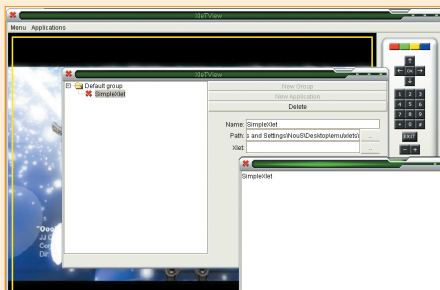
xletview.zip

UTILIZZARE L'EMULATORE PER PROVARE LA NOSTRA APPLICAZIONE

LANCIAMO XLETVIEW

1 Decomprimiamo in una cartella a nostra scelta l'emulatore. Il comando per lanciare il jar del; programma è:

```
java -jar xleTView.jar
```

Se volete compilare il sorgente potete sempre farlo. xletview è opensource.

INIZIALIZZIAMO L'APPLICAZIONE

2 Dopo aver sovrascritto il file *defaultbg.jpg* nella cartella *config* con un'immagine di 720x576 pixel, lanciamo *xleTView* e configuriamo la Xlet da lanciare attraverso il menu *Application -> Manage application.* Bisogna indicare il nome e il path della Xlet scelta. Il wizard riconosce automaticamente le Xlet presenti nella cartella indicata.

VEDIAMO I RISULTATI

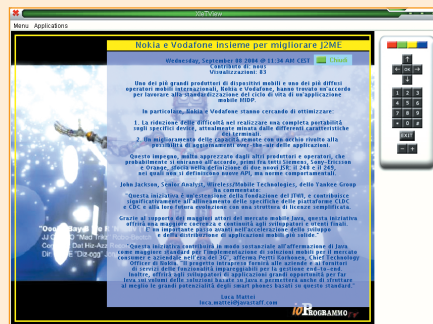
3 Lanciamo la nostra creatura dal menu *Application -> SimpleXlet*. Eventuali messaggi di log, stampati con *System.out.println*, sono visibili nella console da cui abbiamo lanciato xleTView. L'*Application Manager* ha già invocato i metodi *initXlet* e *startXlet*. Difatti la jpg con il menù della nostra applicazione è già visibile.

USIAMO IL TELECOMANDO

4 Premiamo il tasto giallo e vedremo l'anteprima della notizia. Ovviamente il piccolo menù grafico con le possibili opzioni è cambiato con i riferimenti ai possibili stati raggiungibili.

VISUALIZZIAMO LE NOTIZIE

5 Premiamo il tasto rosso e visualizziamo la notizia per intero. La notizia viene visualizzata su uno sfondo di colore solido, così come avevamo previsto in fase di progettazione.

GIOCHIAMO CON LE TRASPARENZE

6 Se premessimo il tasto verde, il pannello diventerebbe invisibile, permettendoci di vedere il video musicale che stiamo seguendo. Ma forse a noi basta che diventi trasparente, vero?

```
import org.havi.ui.HScreen;
import org.havi.ui.HStaticText;
import org.dvb.event.*;
import org.dvb.dsmcc.*;
import java.io.FileInputStream;
import java.io.File;
```

REALI IMPLEMENTAZIONI

Spesso chi si cimenta per la prima volta nella scrittura di una Xlet, rimane meravigliato della semplicità di implementazione su una piattaforma che ci si aspetterebbe più ostica. Questo è sicuramente uno dei vantaggi apportati da Java a questa e a molte altre nuove tecnologie. Di contro c'è la difficoltà iniziale dei vari produttori ad adeguarsi a questa nuova filosofia ed accettare ed implementare correttamente gli standard. Nel primo periodo di vita di una nuova tecnologia come il digitale terrestre, è del tutto normale una scarsa compatibilità dei vari dispositivi con le applicazioni realizzate attenendoci agli standard. A questi problemi, che le aziende impegnate a produrre applicazioni interattive basate su MHP incontrano inevitabilmente, si aggiungono notevoli difficoltà nel deployment del progetto sui decoder. Per l'installazione in broadcast su decoder comuni, infatti, sono necessari costosi server di emissione che solo le televisioni nazionali e qualcuna a livello regionale si possono permettere. Si può ricorrere altrimenti al testing su speciali decoder per sviluppatori, con prezzi dai 1000 dollari in su, che permettono il deployment via cavo da pc. Si tratta però di una situazione momentanea, molto simile a quella riscontrata nei primi periodi di vita di J2ME. L'installazione di applicazioni direttamente attraverso il collegamento ad internet o per mezzo di schede di memoria non è lontanissima.

Luca Mattei

Pagine HTML, Javascript, fogli di stile e la TV è programmata

Sviluppare per Windows Media Center

È l'ultimo nato della famiglia di Windows XP pensato per l'home entertainment, impareremo come programmarlo usando HTML e creeremo una prima applicazione

Utilizza questo spazio per le tue annotazioni

Windows XP Media Center Edition (MCE) 2005 è l'ultimo nato della famiglia di Windows XP. Rappresenta una versione speciale di questo sistema operativo, pensato per l'home entertainment e più in generale per la casa digitale. MCE 2005 è la terza versione ad essere rilasciata, la prima ad essere realmente presentata al grande pubblico. Il lancio mondiale è stato fatto, Italia inclusa, il 12 ottobre scorso con la presentazione di diverse soluzioni hardware basate su questa particolare versione di Windows XP. Con MCE è possibile sfruttare il normale televisore di casa come video per la visualizzazione delle informazioni più disparate: dai DVD fino alle foto scattate nell'ultima vacanza, passando per l'ascolto di CD o MP3, arrivando alla TV o alla radio.

Fig. 1: In evidenza la scheda decoder, cuore di un sistema MCE

ANATOMIA DI UN SISTEMA MCE

Un sistema basato su MCE è tipicamente un computer di ultima generazione, dotato di un particolare hardware in grado di migliorarne le performance. Tipicamente un MCE è dotato di un disco fisso molto capiente, perché tra le sue funzionalità c'è quella di consentire la registrazione di programmi TV in formato digitale, attraverso un'interfaccia molto comoda, che permette di avere, anche per l'Italia, la guida programmi contestuale. Sfruttando Windows Media Player 10, è possibile visualizzare queste registrazioni anche su altri computer in rete o su dispositivi particolari, come i *Portable Media Center*, pensati per estendere le funzionalità di MCE quando gli utenti sono in movimento. Per fare tutto questo è necessario che il sistema sia dotato di una tra le schede certificate, che deve montare a bordo un chip con encoder hardware MPEG 1-2 in tempo reale, così da non gravare sulla CPU per questo genere di operazioni. A parte la scheda di sintonizzazione, che poi ha spesso la funzionalità anche di scheda video e di sintonizzatore radio, un MCE è principalmente un computer basato su architettura Intel, con una versione di Windows XP che ha una "shell" particolare, ottimizzata per le dimensioni e la risoluzione di una TV. Questo in particolare è il grande punto a cui prestare attenzione: una TV normale, una comune 33'', non è in grado di arrivare ad 800x600 dpi di risoluzione mantenendo al contempo una qualità accettabile. A risoluzioni maggiori il basso refresh rate tipico di questo dispositivo si fa sentire, peggiorando la qualità delle immagini.
È per questo motivo che l'interfaccia di MCE è nettamente differente da quella di un computer normale.

LE FUNZIONALITÀ DI MCE

Un MCE è in grado di svolgere diverse funzionalità di base, che possono poi essere aumentate attraverso l'aggiunta di programmi ad hoc. Non appena installato offre queste possibilità:

- **TV:** racchiude le funzionalità legate alla parte TV. Permette sia di visualizzare che di registrare programmi TV, supportando i servizi via antenna, cavo o satellite.
- **Immagini:** consente la visualizzazione sulla TV di immagini, in sequenza e con sottofondi musicali.
- **Radio:** permette di ascoltare la radio FM.
- **DVD:** consente la visualizzazione di DVD video.
- **Video:** racchiude le funzionalità legate alla visualizzazione di video digitali. Il formato DivX non è supportato di default, ma è sufficiente installare il codec per risolvere il problema.
- **Musica:** consente di riprodurre musica nei formato wma (*Windows Media Audio*) e Mp3, tra gli altri. Supporta la visualizzazione di copertine ed è in grado di recuperare le informazioni sul brano da internet
- **Messenger:** è presente un'interfaccia studiata in modo particolare per la tv, che consente di guardare un film ed allo stesso tempo chattare.

Essendo un sistema pensato più per l'intrattenimento che per altri fini, ogni MCE è dotato di un telecomando che consente di comandarne le funzionalità a distanza. Moltissimi sistemi hanno anche una tastiera senza fili, che torna molto utile in alcuni scenari particolari, come la compilazione di form o l'utilizzo di messenger. In definitiva, MCE è un'estensione delle tecnologie Microsoft rivolte al settore della casa digitale.

COME SVILUPPARE PER MCE

Lo sviluppo di applicazioni per MCE segue diverse strade, a seconda delle necessità dell'applicazione da progettare. Come anticipato, la shell di MCE è in grado di fare l'hosting di un'istanza di Internet Explorer, "blindata" in alcuni dettagli. Il primo sistema per sviluppare applicazioni per MCE consiste nella creazione di alcune pagine Web ad hoc, nel normale formato HTML. Si dovrà prestare particolare attenzione a come ricevere l'input dall'utente, dato che il mezzo utilizzato è un telecomando e non un mouse. Mentre con un mouse avrete sempre la classica freccetta che vi indica la posizione, con un telecomando gli elementi cliccabili devono essere evidenziati quando ricevono il focus. Per gestire facilmente questo problema ci viene in aiuto l'SDK che contiene dei file JavaScript studiati per gestire correttamente il focus e il rilascio dei vari elementi.
La seconda tipologia di applicazioni sono gli add-in, che in genere girano all'interno del .NET Framework e comunicano sia con pagine Web sia con MCE stesso. Una terza categoria di applicazioni, in realtà poco diffusa ed utilizzata per le limitazioni che comporta, che è rappresentata dalla costruzione di applicazioni ad-hoc, attraverso WinForm, sempre pensate per il .NET Framework. Dunque per il 90% delle applicazioni che girano su MCE la prima strada, quella della progettazione di pagine HTMML, è la via migliore e consente, tra le altre cose, un deployment delle soluzioni che non impatta minimamente sul sistema e, cosa più importante, non necessita di installazione di file particolari sul computer dell'utente, rendendo anche la sicurezza molto più robusta. Dovremo dotarci dell'SDK che si può trovare su *http://msdn.microsoft.com /mce/*. È gratuito e contiene esempi già utili in disparati scenari, per entrambi i tipi di applicazione.

LE PREROGATIVE E LE LIMITAZIONI DEL MEZZO

Per capire meglio come sviluppare applicazioni per MCE la cosa migliore è senza dubbio cominciare a fare pratica. Come detto, si tratta di normali pagine Web. Ci sono due parametri fondamentali da considerare durante il ciclo di sviluppo. Uno, è che il focus sui vari elementi della pagina è quasi sembre gestito dal telecomando; il secondo è che la nostra periferica di output è tipicamente un televisore, quindi con caratteristiche molto diverse dal classico monitor. Infine, è bene tener presente che le nostre applicazioni saranno nella maggior parte dei casi costituite da pagine HTML con il supporto di ASPX.
In un televisore, la superficie di azione è di circa 800x600 pixel. Ed è bene chiarire subito una forte limitazione: non è possibile lo scrolling diretto della pagina, anche se si può ottenere con alcuni trucchi.

La shell che integra IE utilizza i points (i punti) e non i pixel come unità di misura. Per questo motivo nei CSS non troverete mai "px" ma molto spesso "pt". La scelta è presa in virtù del fatto che i punti non sono legati alla risoluzione scelta e permettono di "scalare" meglio la grandezza dei vari oggetti della pagina in base alla reale risoluzione offerta. Nel caso degli extender o dei Portable Media Center, infatti, la risoluzione è inferiore e spesso pari a 640x480.

INSTALLAZIONE DELLE APPLICAZIONI

Il deployment delle applicazioni è possibile in modalità differenti, ma comunque molto semplici e che richiedono il minimo impatto. La più praticata consiste nella creazione di semplici file XML, che contengono all'interno la lista delle funzionalità che un "programma" supporta. Questi file hanno estensione .mcl e vanno messi nel menu avvio, nella directory "\Accessori\ Media Center\ Programmi Media Center". In questo modo compariranno sotto la voce "Altri programmi" all'interno della shell di MCE. Nel nostro caso il file sarà all'incirca così:

```
<application
   title="Ultime notizie"
   url="http://localhost/mcerss/rss.aspx"
   startImage="http://localhost/
                  mcerss/rss.png"
   thumbnailImage="http://localhost/
      mcerss/rss.png "></application>
```

Title rappresenta il titolo che apparirà nella shell, url l'indirizzo a cui puntare, startImage e thumbnailImages rispettivamente l'immagine ingradita ed in miniatura, visualizzate all'interno della shell. Chiaramente i file dell'applicazione andranno in *\inetpub\wwwroot\mcerss*. Per facilitare l'installazione, esiste poi la possibilità di utilizzare, attraverso javascript, la funzione MediaCenter.RegisterApplication. È ampiamente documentata, anche in questo caso, all'interno dell'SDK e permette di inserire un pulsante alla cui pressione (e senza uscire dalla shell) il programma viene installato. Fatto questo, l'applicazione è pronta per essere utilizzata.

Alla pressione dei tasti sul telecomando, si può associare un'azione corrispondente, attraverso Javascript. Usando SDK di MCE 2005 è sufficiente dotare i tag HTML di un attributo *MCFocusable* impostato su *true*. In tal modo in automatico, attraverso un codice Javascript da includere nella pagina già disponibile nell'SDK, il focus viene passato ai diversi controlli che abbiamo marcato. Ad esempio:

```
<td tabindex=0 id="btn_info" class="button1"
                MCFocusable="True">Informazioni</td>
```

la classe *button1* all'interno di *style.css* è definita come segue

```
.button1 {
behavior:url(Hilite.htc);
background-image: url(Images/
                        Common.Button.Dormant.gif);
z-index: -10;
width: 251; height: 51;
font-weight: bold; font-size: 20pt;
font-family: arial; padding-left: 6;
color: #f2f2f2;
padding-top: 9
}
```

L'attributo *behaviour* consente di definire un comportamento per un oggetto. In questo particolare caso il comportamento degli oggetti di classe *button1* è definito nel file *Hilite.htc*. Se avrete la pazienza di dare uno sguardo scoprirete che:

```
<PUBLIC:ATTACH EVENT="onmouseover" ONEVENT=
                        "mouseOver(event.srcElement)" />
<PUBLIC:ATTACH EVENT="onfocus" ONEVENT=
                        "hilite(event.srcElement)" />
<PUBLIC:ATTACH EVENT="onblur" ONEVENT=
                        "restore(event.srcElement)" />
<PUBLIC:ATTACH EVENT="onclick" ONEVENT=
                        "doClick(window.event.srcElement)" />
<script>
function doClick(item)
{  // this function is necessary to update the oCurFocus
      variable, in case the mouseover fails to do that
   // then it calls the doSelect function
   oCurFocus = item
   doSelect() }
</script>
<script src="Hilite.js" type="text/javascript"></script>
```

Hilite.htc importa alcune funzioni dal file JavaScript *Hilite.js*, in particolare quelle utilizzate per gestire gli eventi *mouseOver*, *hilite* e *restore*. Per quanto riguarda l'*Onclick*, definisce una funzione *doClick* che non fa altro che richiamare una funzione *doSelect* la cui implementazione è lasciata al programmatore. È ovvio infatti che saremo noi a dover programmare cosa succede quando un utente clicca su un bottone.

SUL WEB

Procurasi l'SDK è semplice. Basta puntare il proprio browser all'indirizzo http://msdn.microsoft.com/mce/ dove ci sono link ad altre risorse interessanti e alla documentazione in linea, in formato HTML. Ci sono poi community molto attive, tra cui sicuramente la più importante è TheGreenButton: www.thegreenbutton.com

POSIZIONARE GLI ELEMENTI

Una parte importante del sistema è rappresentata dall'esatta collocazione degli elementi all'interno della pagina. Per fare questo in genere si utilizza il posizionamento assoluto attraverso CSS, dato che gli elementi della pagina hanno quasi sempre l'attributo *overflow* sul valore *hidden*, che in pratica nasconde la dimensione dell'oggetto oltre quella specificata nell'attributo *width*. In questo modo è possibile demandare le funzioni di scrolling a codice javascript da includere (ma già disponibile nell'SDK) che si occupa di ottimizzare tutto al meglio.

NOTA

COSTI

Il sistema Windows Media Center che vedete in figura ha un costo di € 1.800. È dotato di una CPU Intel P4 540 da 3,2 GHZ, 512MB di RAM DDR, un HD Maxtor da 200 GB.

Fig. 3: Nell'immagine qui a fianco è possibile vedere un media center. In questo caso si tratta di un modello prodotto da Frael. In questa foto non collegato alla TV

CONCLUSIONI

Lo sviluppo di soluzioni per il Media Center è un argomento per certi versi inesplorato. Allo stato attuale è possibile creare applicazioni di un certo impatto con relativamente poca difficoltà. Con questo articolo diventerà più semplice muovere i primi passi verso lo sviluppo di un genere di applicazioni che nei prossimi anni, avrà un ruolo importante nel segmento casalingo. Per toccare con mano MCE è possibile installarlo in una macchina virtuale, all'interno di VirtualPC o VMWare, anche se ovviamente non tutte le funzionalità saranno disponibili.

Daniele Bochicchio

LA NOSTRA APPLICAZIONE: LE NOTIZIE VIA RSS

MCE è un dispositivo che sta benissimo in un salotto. Per questo motivo, un comodo servizio di lettura di notizie in formato RSS *(Rich Site Summary)* è quello che ci vuole per rendere più comoda la consultazione degli ultimi fatti accaduti nel mondo. Per il nostro esempio ci soffermeremo, più che sulla parte funzionale vera e propria, che si basa su una soluzione ASP.NET, sulle problematiche proprie di MCE. Nel nostro esempio utilizzeremo un file di nome *rss.aspx* che utilizza una logica molto semplice

1 PARAMETRI DI BASE

```
<SCRIPT RUNAT="SERVER" Language="C#">
void Page_Load()
{
   AddSource(ConfigurationSettings.AppSettings[
                                          "rss"]);
}
```

Al caricamento della pagina viene richiamata una funzione

```
AddSource(ConfigurationSettings.AppSettings["rss"]);
```

Il parametro passato a tale funzione viene recuperato dal file *web.config*. Nel nostro caso corrisponderà all'URL da cui vogliamo recuperare le informazioni in formato *rss*.

2 RECUPERA LE INFORMAZIONI

```
void AddSource(string Url) {
   // caricamento del documento XML attraverso
                                    XmlDocument
   XmlDocument myXmlDoc = new XmlDocument();
   myXmlDoc.Load(Url);
   // caricamento dell'XSLT
   XslTransform myXSLT = new XslTransform();
   myXSLT.Load(Server.MapPath("style.xsl"));
   // imposto le proprietà sul file
   myXML.Transform = myXSLT;
   myXML.Document = myXmlDoc; }
</SCRIPT>
```

Il parametro *Url* contiene l'indirizzo dell'URL da parserizzare. Le informazioni ricevute vengono trasformate tramite XSL. La definizione del file XSL è contenuta nel file *style.xsl*

3 PRESENTAZIONE A VIDEO

```
<html><head>
<title>Rss su Media Center</title>
<link rel="STYLESHEET" type="text/css" href= "style.css">
<script src="BasicFunctions.js" type="text/javascript">
                                              </script>
<script src="Scrolling.js" type="text/javascript">
                                              </script>
<script src="MoveFocus.js" type="text/javascript">
                                              </script>
<script language="JScript" id="clientEventHandlersJS">
   function pageLoadFunctions()
   {  setBGColor("#666666");
      checkSVP();
      setCounter();
      setArray();
      startFocus(); }
   // funzione richiamata per associare le azioni sui pulsanti
   function doSelect()
   {  // vai all'url
```

```
      if (oCurFocus.id.indexOf('goto_url_')>=0)
      { document.location = 'read.aspx?url=' +
              oCurFocus.id.replace('goto_url_', ''); }
       switch(oCurFocus.id)
       { case "btn_info": // ID of button
          {   document.location = 'info.aspx';
              break; } } }
</script></head>
```

Come vedete vengono inclusi tutti i file JavaScript e i fogli di stile che servono al buon funzionamento dell'applicazione. Viene lasciata al programmatore la definizione della funzione *doSelect()*. Se l'utente cliccherà su una news, verrà richiamato il file *read.aspx*, cui verrà passato un parametro necessario a leggere la news corrente. Se invece l'utente cliccherà sul bottone info, verrà richiamata la pagina *info.aspx*

4 L'INTERFACCIA GRAFICA

```
<body id="body" MCFocusStart="btn_info" onload=
                  "pageLoadFunctions()" onkeydown=
            "onRemoteEvent(window.event.keyCode)">
<bgsound id="btnSound"/>
<!-- Shared Viewport -->
<span style="position: absolute; top: 0; left: 0;
                                   height: 100%;">
   <table style="position: absolute; top: 0; left: 0;
     height: 100%;" cellspacing="0" cellpadding="0">
       <tr><td valign="bottom" height="100%">
       <span id="SVP" style="width: 308;
                             height: 216;
                             vertical-align: bottom"
                             MCFocusable="true">
       </span>
    </td></tr>
    </table>
</span>
<!-- Shared Viewport -->
[...]
<!-- pulsanti dx -->
<table id="listTable" border="0" cellpadding=
         "0"cellspacing="3" style="position: absolute;
                                top: 97; left: 10;">
    <tr><td tabindex=0 id="btn_info" class="button1"
        MCFocusable="true">Informazioni</td></tr>
</table>
<!-- pulsanti dx -->
<!-- contenuto pagina -->
<asp:xml id="myXML" runat="server" />
</body></html>
```

Viene definita la *SharedViewPort*, ovvero la finestra in basso a sinistra che mantiene le immagini della TV o di altre applicazioni, viene disegnato il bottone *"Informazioni"* a sinistra e infine il contenuto della pagina viene associato all'oggetto *myXML* riempito in precedenza con una trasformazione XSLT.

LA FUNZIONE DOSELECT()

Abbiamo usato un banale trucco. Viene definito, all'interno dell'ID dell'immagine, una parte comune composta da "*'goto_url_*" ed una parte variabile fatta dell'URL vero e proprio, in modo da facilitare il *redirect* alla pagina che si occuperà di mostrare la notizia nel dettaglio.

LA PAGINA DI DETTAGLIO

La visualizzazione del dettaglio della notizia è effettuato semplicemente aprendo l'url all'interno di un iframe. Purtroppo in casi come questi si può toccare facilmente con mano la grossa limitazione che la bassa risoluzione della TV porta con sé. Ad ogni modo, la pagina di dettaglio contiene un iframe, con un po' di javascript particolare preso sempre dall'SDK, che inserisce un iframe molto lungo, bypassando le funzionalità di scrolling che offre l'SDK per fare in modo che le protezioni di Javascript quando ci sono url esterni non diano problemi. È davvero essenziale, e questo esempio lo testimonia, che le applicazioni vengano pensate e progettate per questo diverso sistema, pena una qualità delle stesse davvero molto bassa.

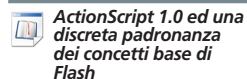
Sviluppare applicazioni Flash per cellulari e palmari

Macromedia Flash e il Mobile

Impareremo a costruire applicazioni Action Script per dispositivi mobili riconoscendo la piattaforma e scambiando dati con il Web L'obiettivo è visualizzare gli spettacoli che si terranno in città!

di Giorgio Natili

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

Grazie al peso ridotto, alle sue capacità di gestire animazioni vettoriale e media esterni, Flash è diventato uno standard de facto per la visualizzazione di contenuti multimediali attraverso Internet. Esamineremo le problematiche da considerare quando si produce un'applicazione per dispositivi mobili e gli aspetti da tenere presente quando si progetta per Pocket Pc. Sviluppare applicazioni per dispositivi mobili è un'attività, per certi punti di vista, molto più delicata e complessa dello sviluppo di software utilizzabile su desktop o via Internet attraverso strumenti tradizionali come i PC. Un aspetto importante da considerare è che, se nel Web abbiamo differenti browser che girano su differenti piattaforme a differenti risoluzioni, i dispositivi mobili hanno degli schermi di forma molto differente a seconda della ditta che li produce, pensiamo ad esempio al Nokia Communicator che ha uno schermo largo almeno tre volte un qualsiasi Pocket Pc e, comunque, di dimensioni più piccole rispetto alle classiche risoluzioni dei monitor per i quali siamo abituati a progettare.

OPERAZIONI PRELIMINARI

La prima applicazione che svilupperemo sarà destinata ai cosiddetti dispositivi Pocket Pc su cui girano diverse versioni del sistema operativo Microsoft Pocket PC (*http://www.microsoft.com/ windowsmobile/pocketpc/ppc/default.mspx*) e per i quali è disponibile la versione 6 del Flash Player. Macromedia mette a disposizione, oltre ad una serie di template già inclusi in Flash, un pacchetto gratuito di supporto per gli sviluppatori *(FP6_ PPC_DEV_KIT)* scaricabile direttamente dall'indirizzo *www.macromedia.com/devnet/devices/pocket_pc.html*. Il pacchetto contiene degli esempi, alcune guide ed una serie di *components* studiati appositamente per lo sviluppo su palmari. Li trovate nel kit di sviluppo nella cartella components all'interno di una serie di *.fla* (uno che li contiene tutti e un file per ogni component) e per installarli è sufficiente copiarli nella directory *C:\Programmi\Macromedia\Flash MX 2004\lingua\First Run\Components*, chiudere Flash e riaprirlo. Da questo momento li troverete a disposizione nel pannello dedicato ai components *(CTRL + F7)* accessibile dal menu *Window > Development Panels > Components*. Sempre nei file distribuiti per lo sviluppo è a disposizione una guida in *.pdf* ed una serie di template dedicati ai Pocket Pc di produzione più recente da copiare all'interno della cartella *C:\Programmi\Macromedia\Flash MX 2004\lingua\Configuration\Templates.*

LA NOSTRA PRIMA APPLICAZIONE PER POCKET PC

Apriamo Flash e iniziamo a sviluppare la nostra semplice applicazione che eseguirà alcune facili operazioni: verificherà se viene visualizzata attraverso un Pocket Pc, controllerà se il dispositivo è connesso ad Internet, esaminerà le sue caratteristiche, come ad esempio la capacità di riprodurre contenuti video e audio, la ram disponibile, la versione del sistema operativo, ecc. e condurrà l'utente all'inserimento del nome e del cognome in modo da riconoscerlo durante i successivi accessi. L'utente, al completamento di queste operazioni, potrà accedere alla lista degli spettacoli che si terranno in città quel giorno. Selezioniamo dal menu *File > New* e andiamo scegliere uno dei template per *Mobile Devices*, ad esempio l'ultimo della serie IPAQ – 54 X0, che utilizzeremo come base per sviluppare un'applicazione per Pocket Pc accessibile dal browser. Selezioniamo lo stage e, nel pannello

Properties, noteremo che le impostazioni rispecchiano quelle standard di Flash MX: ActionScript 1.0, 12 FPS e sfondo bianco. Le dimensioni dello stage sono di 240 X 268 pixel, modifichiamole e impostiamo i valori *220* e *235* rispettivamente per la larghezza e per l'altezza, in questo modo terremo conto delle barre degli strumenti del browser) e abbassiamo a 6 FPS la frequenza di riproduzione del nostro filmato per mantenere basso l'impegno della cpu.

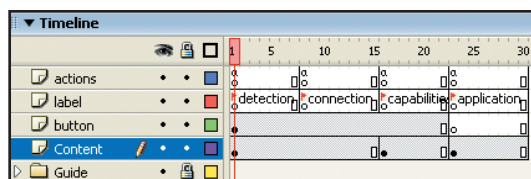



Fig. 1: ***Organizzazione della Timeline***

Apriamo il menu *File* > *Publish Setting* abilitiamo la pubblicazione della pagina *.html* e, selezionando prima il tab che porta alle opzioni di pubblicazione relative a questo formato, scegliamo dal menu a tendina posto in alto l'opzione *Flash For Pocket Pc*. Torniamo nell'interfaccia e cominciamo a organizzare la nostra timeline inserendo un livello per le azioni, uno per le etichette dei fotogrammi, uno per i pulsanti di navigazione e uno per i contenuti visualizzati in ogni step dell'applicazione (**Figura 1**).

1 RILEVAMENTO PIATTAFORMA - Inseriamo, nel primo fotogramma chiave della nostra timeline un campo di testo statico contenente il testo *"Rilevamento Piattaforma..."*, una campo di testo dinamico al quale assegniamo il nome istanza *detect_txt* e un pulsante che invece avrà come nome istanza *next_btn* potete crearlo voi stessi o prelevare direttamente dalle librerie comuni. Apriamo il pannello delle azioni e, selezionando un fotogramma chiave nel livello *Actions*, inseriamo questo semplice script. Come prima cosa, blocco il ridimensionamento dello stage e dichiaro la funzione che verificherà se il Flash Player in cui gira l'applicazione è per Pocket Pc. La funzione *beginsWith* esegue un confronto tra due stringhe, la stringa "*WINCE*", che indica che il Player è per Pocket Pc, e la stringa memorizzata nella variabile player che contiene la versione del Flash Player.

```
Stage.scaleMode = "noScale";
function beginsWith(s, comparison){
    return(s == comparison.substring(0, s.length));}
// Recupero la versione del Player
var player = getVersion();
```

Sono pronto per verificare se l'applicazione è visualizza su Pocket Pc e popolo il campo di testo dinamico *detect_txt*. Ne cambio il colore in base al risultato restituito dalla funzione *beginsWith*.

```
if(beginsWith("WINCE", player)){
```

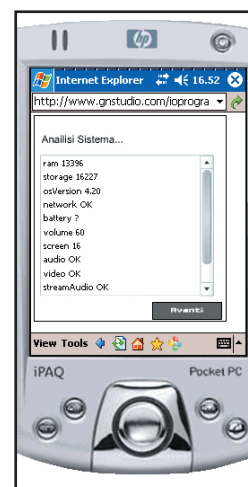



Fig. 2: ***Verifica delle caratteristiche del dispositivo***

PIANIFICARE E PROGETTARE UN'APPLICAZIONE

Prima di cominciare a scrivere il codice di un'applicazione o di buttare giù un'interfaccia, è d'obbligo esaminare a fondo le problematiche che si prevede di incontrare e analizzare le possibili soluzioni. Quando vogliamo utilizzare Flash per un'applicazione distribuita su dispositivi mobili dobbiamo progettarla in modo che sia il più leggera possibile senza impegnare troppo la CPU del dispositivo. Ecco un elenco di sette regole d'oro:

1. **Ponete molta cura nell'ottimizzazione delle immagini che utilizzate: è consigliabile evitare immagini troppo dettagliate e transizioni molto complesse.**
2. **Gli effetti che più rallentano l'applicazione sono: le interpolazioni di forma, zoom improvvisi, un elevato numero di interpolazioni di movimento simultanee, animazioni di scritte e visualizzare troppi simboli contemporaneamente.**
3. **Quando scegliete di utilizzare immagini bitmap, dovete cercare di ottimizzarle al massimo avendo cura di ridimensionarle prima di importarle in Flash.**
4. **Scegliere di usare immagini vettoriali comporta un minor impatto sulle dimensioni dei file (sono formate solo da equazioni matematiche). Visto che la rappresentazione è frutto di una elaborazione, i processori dei palmari possono "soffrire" carichi di lavoro ingenti.**
5. **Lavorando con i palmari occorre rivalutare un formato di compressione audio ben noto agli sviluppatori Flash e che risulta molto meno impegnativo per il processore rispetto agli mp3: Adaptive Differential Pulse Code Modulation (ADPCM). I suoni importati possono essere compressi direttamente dalla libreria di Flash selezionandoli e utilizzando il tasto destro per accedere alla voce *Export Settings* presente nel menù contestuale.**
6. **Uno dei più grossi problemi incontrati dagli sviluppatori ActionScript, risolto con il Palyer 7 $ ma (al momento non disponibile per i dispositivi mobili), è sempre stato quello di visualizzare caratteri di dimensioni molto piccole senza che l'antialiasing di Flash li visualizzasse sgranati e ben poco leggibili. Per risolvere questo problema si possono utilizzare i pixel font (ne trovate varie da scaricare sul sito http://www.miniml.com). Trattasi di caratteri bitmap che non incidono sulla dimensione dinale del file. I pixel font devono essere esportati nel filmato e devono essere posizionati su coordinate intere: (0,5) e non (0, 5.5).**
7. **Dividete il progetto in più file, contenenti sia grafica che dati, e caricateli a seconda delle richieste effettuate dall'utente finale. Per caricare file swf differenti si può tranquillamente utilizzare l'azione *loadMovieNum,* mentre quando si ha a che fare con i dati si può sia usare il classico *loadVariables.***

COMPONENTS

I components, per chi non lo sapesse, sono stati introdotti in Flash Mx e sono gli eredi degli Smart Clip; sono dei clip filmato contenenti il codice necessario per eseguire funzionalità avanzate come visualizzare dei dati in una ListBox, gestire scelte dell'utente attraverso dei RadioButtons, ecc. La versione per Pocket Pc è molto più leggera e meno pesante da eseguire ed è quindi adatta ad applicazioni studiate per palmari e dispositivi mobili in genere.

TECONOLOGIE WIRELESS E MOBILE

Access Point: Punti di accesso, all'interno di un infrastruttura o netWLan (rete senza fili) fungono da ricevitori/trasmettitori fissi con i dispositivi mobili che comunicano. Possono essere usati semplicemente come ripetitori di segnale, o come elementi di interfaccia tra mondo senza fili (wireless) e mondo cablato svolgendo funzioni analoghe ad un bridge o router.

```
  detect_txt.textColor =  0x333333;
  detect_txt.text = "Pocket PC Rilevato!\
                  nClicca su avanti per proseguire...";
} else {
  detect_txt.textColor = 0xff0000;
  detect_txt.text = "Attenzione! \nPocket PC NON
                                        rilevato.";
}
```

Associo alla pressione del pulsante *next_btn* le azioni necessarie per disabilitalo e avviare la riproduzione del filmato

```
next_btn.onPress = function(){
  play();
  this.enabled = false; }
```

Interrompo la riproduzione del filmato

```
stop();
```

2 RILEVAMENTO DELLA CONNESSIONE - Il secondo step prevede che l'applicazione rilevi se il dispositivo è connesso a Internet, semplicemente verificando il corretto caricamento del file *connection.txt* contenente un messaggio memorizzato nella variabile status attraverso l'oggetto *LoadVars*. Svuoto il campo di testo dinamico, riabilito il pulsante, dichiaro una variabile che contiene il *txt* da chiamare, *urlToTest*, e creo una nuova istanza dell'oggetto *LoadVars*

```
detect_txt.text = "";
next_btn.enabled = true;
var urlToTest = "connection.txt";
var detectConnection = new LoadVars();
```

Gestisco il caricamento del file e, se è caricato correttamente, visualizzo il messaggio contenuto nel *txt*, in caso contrario indico all'utente che non è connesso a Internet

```
detectConnection.onLoad = function(success){
  if(success){
    detect_txt.textColor = 0x333333;
    detect_txt.text = this.status;
  } else {
    detect_txt.textColor = 0xff0000;
    detect_txt.text = "Il dispositivo non è connesso a
                                     Internet..."; }
}
```

Carico il file memorizzato nella variabile *urlToTest*

```
detectConnection.load(urlToTest);
stop();
```

3 ANALISI DEL SISTEMA - Nel terzo step utilizzeremo una *ListBox*, che trascineremo sullo stage dal pannello dei components e alla quale assegneremo il nome istanza *system_lb*, per visualizzare tutte le informazioni recuperate dall'oggetto *System.capabilities*

```
next_btn.enabled = true;
```

Utilizzo l'oggetto *System.capabilities* per recuperare informazioni circa il dispositivo su cui sta girando l'applicazione

```
var ramAvailable = System.capabilities.ramAvailable;
var storageAvailable =
                  System.capabilities.storageAvailable;
var osVersion = System.capabilities.osVersion;
var username = System.capabilities.username;
var network = System.capabilities.network;
var battery = System.capabilities.battery;
var volume = System.capabilities.volume;
.........................................................................
```

Popolo il component *ListBox* al quale ho assegnato il nome istanza *system_lb* utilizzando per formattare i dati in maniera che siano più facilmente intelligibili la funzione *converCapData(rw)*;

```
system_lb.addItem("ram " +
converCapData(ramAvailable));
system_lb.addItem("storage " +
converCapData(storageAvailable));
system_lb.addItem("osVersion " +
converCapData(osVersion));
.........................................................................
```

Con funzione *converCapData* gestisco i risultati ottenuti dall'oggetto *System.capabilities* rendendoli comprensibili per l'utente finale

```
function converCapData(rw){
  var stat = "";
  switch(rw){
    case true:
      stat = "OK";
    break;
    case false:
      stat = "NO";
    break;
    case undefined:
      stat = "?";
    break;
    default:
      stat = rw; }
  return stat; }
stop();
```

4 APPLICATION - Nel quarto step verificheremo se l'utente ha già utilizzato questa applicazione o meno, gestendo la riproduzione del clip filmato *application_mc* contenente i tre passi finale della nostra utility per visualizzare le schede degli eventi mondani di alcune città italiane. Il Flash Player 6 per Pocket Pc, così come quello disponibile per computer fissi, ci offre la possibilità di archiviare dei dati all'interno del dispositivo mobile che sta visualizzando l'applicazione attraverso gli *SharedObject.* Gli *SharedObject* vengono definiti i cookie di Flash, dal momento che hanno la stessa funzione dei cookie tradizionali, ovvero salvare informazioni sul computer dell'utente per poterle recuperare in un momento successivo. Ma di diverso da questi, hanno la posizione in cui vengono salvati, determinate cartelle di sistema, le

modalità di utilizzo e recupero, e il sistema di sicurezza relativo all'accessibilità delle informazioni stesse. Ecco il codice necessario recuperare (o creare, se inesistente) uno *SharedObject* per memorizzare nome e cognome dell'utente

```
my_so = SharedObject.getLocal("userdata");
```

Se sono presenti i dati, e quindi non è il primo accesso, sposto la testina di riproduzione al fotogramma *"choice"* del clip *application_mc*

```
if(my_so.data.userName != undefined &&
my_so.data.userSurname != undefined){
   application_mc.gotoAndStop("choice"); }
stop();
```

Il clip filmato *application_mc* contiene tre step separati: il primo, etichetta fotogramma *firsttime,* consente l'inserimento de propri dati all'utente che accede la prima volta all'applicazione che automaticamente lo riconoscerà all'accesso successivo; il secondo, etichetta fotogramma choice, che permette di scegliere una città da una *ListBox;* il terzo, etichetta fotogramma display, caricherà, in base alla scelta eseguita nello step precedente, un file *.swf* esterno contenuto nella cartella *__events__.*

```
Step 4 Application  > Fotogramma firsttime
```

Attraverso il pulsante di registrazione *register_btn* svuoto il campo di testo dinamico e memorizzo in due variabili il testo contenuto nei campi di input

```
register_btn.onPress = function(){
   this._parent.message_txt.text = "";
   var name = name_txt.text;
   var surname = surname_txt.text;
```

Se non sono vuoti e se non contengono i valori di default memorizzo il loro contenuto nello *SharedObject* e passo alla fase successiva, altrimenti se i dati non sono corretti mostro un messaggio di errore

```
   if(name != "" && surname != "" && name !=
            "nome..." && surname != "cognome..."){
      this._parent._parent.my_so.data.userName = name;
      this._parent._parent.my_so.data.userSurname =
                                              surname;
      this._parent._parent.my_so.flush();
      gotoAndStop("choice");
   } else {
      this._parent.message_txt.text = "Inserire nome e
                                             cognome!";
   }
}
```

La prima volta che viene selezionato, svuota il campo di testo di input *name_txt*

```
name_txt.onSetFocus = function(){
   if(this.text == "nome...") this.text = ""; }
.........................................................................
stop();
Step 4 Application > Fotogramma choice
```

Genero il messaggio di benvenuto con i dati inseriti dall'utente

```
message_txt.text = "Benvenuto " +
        this._parent.my_so.data.userName + " " +
     this._parent.my_so.data.userSurname + " nella
 nostra applicazione per consultare gli eventi monda-
                 ni delle principali città italiane!";
```

Disabilito il pulsante di avanzamento

```
next_btn.enabled = false;
```

Imposto la funzione da eseguire quando viene selezionato una voce della ListBox che contiene le città

```
cities_lb.setChangeHandler("activeNext");
```

Funzione che riabilita il pulsante di avanzamento associata alla selezione della ListBox

```
function activeNext(){
         next_btn.enabled = true; }
```

Recupero i dati contenuti nell'elemento selezionato e passo allo step finale

```
next_btn.onPress = function(){
   cityID = cities_lb.getSelectedItem().data;
   cityName = cities_lb.getSelectedItem().label;
   gotoAndStop("display"); }
stop();
Step 4 Application > Fotogramma display
```

Popolo il campo di testo dinamico

```
event_txt.text = "Evento del giorno a " + cityName;
```

Carico un *swf* esterno associato all'evento scelto

```
loadMovieNum("__events__/" + cityID + ".swf", 1);
```

Recupero la data odierna

```
var today = new Date();
dataDate = today.toString().split(" ")
```

Popolo il campo di testo in cui è possibile visualizzare i dati

```
eventDetails_txt.text += cityName + " " + dataDate[
     2]  + " " + dataDate[1] +  "\n"; eventDetails_
       txt.text += "Al momento non sono disponibili
          ulteriori dettagli sull'evento in questione...";
```

Associo al pulsante *back_btn* le azioni necessarie per scaricare il file *.swf* caricato e per tornare allo step precedente

```
back_btn.onPress = function(){
         unloadMovieNum(1);
         gotoAndStop("choice"); }
```

L'applicazione è ora pronta per girare su dispositivi mobili che utilizzano il sistema operativo Microsoft Pocket Pc!

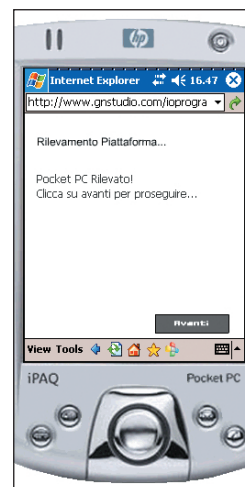



Fig. 3: ***Il rilevamento della piattaforma ha avuto successo!***

BLUETOOTH

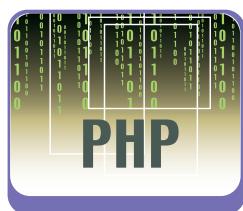
È una tecnologia di interconnessione wireless low-power (mWatt), in grado di far "comunicare" dispositivi elettronici come i telefoni, stereo, notebook,computer, pda fino ad un massimo di 16 dispositivi, attraverso onde radio a basso raggio emesse da alcuni trasmettitori presenti all'interno di questi dispositivi.

IRDA

Infrared Device Application, standard di interconnessione dati tramite infrarossi bidirezionale point-to-point tra dispositivi posizionati in visibilità reciproca (LoS, line of sight) con range .

Usare SQLITE, il nuovo DB integrato con PHP5

Database senza Database

Tutte le novità di SQLITE, il database incluso di default in PHP5 che, per progetti di medie dimensioni, prende il posto di MySQL. Molto comodo anche per la portabilità su ogni tipo di macchina

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
Principi di PHP

Software
Visual Studio .NET 2003

Impegno

Tempo di realizzazione

Dopo anni di attesa PHP5 ha fatto il suo ingresso in scena. Le novità sono profonde e radicali. Il paradigma ad oggetti è ormai alla base del nuovo PHP. XML è una realtà fortemente presente. Il fortissimo legame che univa PHP a MySQL è stato in qualche modo indebolito.
In PHP5, nelle versioni per Windows, il supporto a MySQL non è più compilato all'interno dei binari. L'integrazione con MySQL viene gestita ora con il classico meccanismo delle estensioni, ovvero è necessario caricare *mysql.dll* dal *php.ini* per potere accedere alle funzionalità tipiche di MySQL. Chiaramente, nei sistemi Unix questo discorso si avverte meno poiché la compilazione è demandata, quasi sempre, al sistemista. La cosa interessante è che, attualmente, il database integrato con PHP5 è SQLITE, che come vedremo offre delle novità architetturali molto interessanti rispetto a MySQL.

LA LOGICA DI SQLITE

La particolarità più interessante di SQLITE è che non è da considerare come un software separato. Non ha bisogno di essere installato. Non gira su una porta particolare. Non ha una shell di gestione separata come MySQL.

E allora come funziona? Molto semplicemente in PHP5 è integrata la libreria SQLITE, che include funzioni che consentono la creazione e la gestione di file su disco, che costituiscono la base di dati. In sostanza creare un database SQLITE è concettualmente identico a poter scrivere un file sull'HD.
Questo significa che tutta la sicurezza del database non viene più gestita da un server DB ma direttamente dai permessi impostati su disco.
Tanto per essere chiari, mentre prima eravamo abituati a creare un utente MySQL con permessi speciali di accesso a singoli db, o a porzioni di db, tutto questo non esiste più. Il file contenente i dati viene scritto su disco e la sicurezza è demandata al sistema.
Se questo ha degli svantaggi in termini di gestione dei permessi sulle singole tabelle e sul database stesso, è evidente che invece ha degli straordinari vantaggi in termini di flessibilità del sistema.
Infatti, non c'è nessuna necessità di avere abilitato il supporto a MySQL per gestire i database. Tutto quello che vi serve è semplicemente PHP5.

TUTTE LE CARATTERISTICHE DI SQLITE

- **Supporta le transazioni. In MySQL il supporto alle transazioni è arrivato solo di recente.**
- **Non ha bisogno di nessuna configurazione per essere usato**
- **L'intero database viene scritto in un file su disco**
- **Supporta database di dimensione fino a 2 terabytes**
- **Implementa quasi interamente lo standard SQL92**
- **La portabilità è assicurata**
- **La dimensione delle stringhe e dei campi Blob è limitata solo dalla dimensione della memoria disponibile**
- **Il codice sorgente è disponibile**

Il paragone è con i classici file *.MDB* di Access. Non esiste per questi file un vero server di database per la gestione, ma tutto è demandato al sistema. Chiaramente SQLite offre funzionalità molto diverse rispetto al formato Access, ma il principio rimane invariato.

I TIPI DI DATI

Udite udite, SQLite è scarsamente tipizzato. Potete creare una tabella o inserire dati in una colonna non tenendo in considerazione il suo tipo di dato. Nella versione di PHP utilizzata per scrivere questo articolo, la 5.0.0 è incluso il supporto a SQLite 2.8.14.
La manualistica di SQLite recita che nelle versioni della serie 2 tutti i dati vengono salvati come se fossero un normale campo di tipo *Char*. Questa è una caratteristica voluta e ricercata dai programmatori di SQLite e non una dimenticanza. L'idea è che un database dovrebbe servire a immagazzinare e fornire dati e non dovrebbe occuparsi di che tipo sono questi dati. La gestione dei tipi di dati è demandata al programmatore. In realtà questo è un concetto caro anche al buon PHP che è rimasto forse l'unico linguaggio scarsamente tipizzato del variegato mondo della programmazione. L'assenza del suporto ai tipi di dati, implica che siano ammissibili operazioni del tipo:

```
CREATE TABLE esempio (a,b,c);
```

nonostante ciò, è ancora possibile creare una tabella con la sintassi seguente

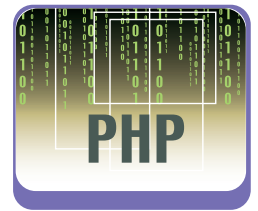
```
CREATE TABLE esempio2(
  a VARCHAR(10),
  b NVARCHAR(15),
  c TEXT,
)
```

Questo tipo di sintassi non aggiunge nulla alla sostanza. In realtà i campi rimarranno privi di tipo e potrete gestirli come meglio credete, ma è utile al programmatore come una sorta di post it che indica cosa dovrebbero contenere quei campi.
A quanto detto sopra, fa unica eccezione la seguente dichiarazione

```
CREATE TABLE Redazione (
   id INTEGER primary KEY
)
```

che dichiara un campo come *"Intero"* e lo gestisce come autoincrementale. In sostanza, un campo *INTEGER PRIMARY KEY* implementa quella che è normalmente una chiave unica. Qualunque inserimento di un valore *"Null"* all'interno di un campo dichiarato in questo modo, viene sostituito con un numero che è di una un'unità più grande dell'ultimo inserimento effettuato.
Infine, è importante sottolineare che qualche nozione di tipo di dato può essere fornita quando si deve effettuare una comparazione allo scopo di determinare un ordinamento.
Perciò è possibile dichiare un tipo come *text* o *numeric*.

UNA GROSSA LIMITAZIONE

Poichè SQLite utilizza un unico file di testo per immagazzinare i dati, appare evidente che si potrebbero avere problemi se più processi accedessero in scrittura al file in questione. Per questo motivo in SQLITE è supportato l'accesso in lettura da più utenti contemporaneamente, ma non quello in scrittura.
Quando un utente accede in scrittura al database, ad esempio per eseguire una funzione di inserto, il database viene bloccato con un *Lock*, e se un secondo processo dovesse tentare una *insert* sul file in questione mentre questo è ancora "loccato" il processo di scrittura

ESISTONO DEI FRONT END GRAFICI PER LA GESTIONE DI DATABASE IN FORMATO SQL LITE?

Si, ne esistono. Uno molto interessante è il SQL Lite Manager reperibile all'indirizzo http://sqlabs.net/index.shtml?sqlitemanager È possibile creare query, database e tabelle. Le possibilità ovviamente sono limitate in quanto SQLite è per sua natura un prodotto da essere usato in modo programmatico.

MA QUANTO È VELOCE SQLITE?

Sono stati effettuati alcuni test, utilizzando come base un Athlon da 1.6GHZ, 1 GB di RAM e un disco IDE. Il sistema operativo era una Redhat Linux 7.2. I database concorrenti erano PostgresSQL 7.1.3, e MySQL 3.23.41.
In questa configurazione SQLite si è mostrato mediamente 10 o 20 volte più veloce di Postgres per le operazioni più comuni. Ha avuto invece piu o meno le stesse prestazioni di MySQL per le stesse operazioni. Vi riportiamo qualche esempio:

	PostgreSQL:	MySQL:	SQLite 2.7.6:
INSERT in una transazione	4.900	2.184	0.914
INSERT in una tabella indicizzata	8.175	3.197	1.555
SELECT su una tabella non indicizzata	3.629	2.760	2.494
SELECTs su una tabella indicizzata index	4.614 :	1.270	1.121
UPDATE in una tabella non indicizzata	1.739	8.410	0.637
UPDATE in una tabella indicizzata	18.797	8.134	3.520

Il test intero con le relative query è disponibile all'indirizzo http://sqlite.org/speed.html

verrebbe ritardato fino al momento in cui il primo processo non termina il suo compito e il file viene liberato. Il lock avviene tramite la scrittura su disco di un file apposito. Ogni volta che un processo tenta un'operazione di scrittura, verifica prima la presenza di questo file. Se non trova nessun file di lock dà il via per l'esecuzione dell'operazione. In server di database destinati a un uso meno domestico, il lock di un dato è più raffinato, in quanto è possibile ad esempio "loccare" la singola tabella e non l'intero database.

SUL WEB

Ci sono parecchi riferimenti per la documentazione relativa a SQLlite. Non tutti sono utilizzabili con PHP, dato che in realtà SQLlite può essere utilizzato anche con C++, con TCL, con Python, con tutta una serie di linguaggi. In ogni caso:

Un buon wiki
http://www.sqlite.org/cvstrac/wiki

La documentazione ufficiale
http://sqlite.org/docs.html

La sintassi
http://sqlite.org/lang.html

IL CODICE COMMENTATO

In questo articolo vi abbiamo proposto un semplice esempio di integrazione PHP – SQLITE. L'idea è mostrarvi brevemente tutte le funzioni più interessanti del DB.
Nel **passo 1**, viene creato un database dal nome *ioProgrammo.db.* Senza nessun tipo di setup o installazione, ma semplicemente avendo installato PHP5, noterete che la prima volta in cui viene richiamata la pagina contenente il codice definito nel **passo 1**, verrà creato nel vostro Hard Disk il file *ioProgrammo.db.* È qui dentro che finiranno tutti i vostri dati. Al successivo riavvio della pagina, il file in questione non viene sovrascritto e nessun *Warning* viene generato. Semplicemente l'operazione viene ignorata.
Avrete anche notato che abbiamo utilizzato la programmazione a oggetti piuttosto che quella procedurale. Per quanto riguarda PHP5 il nostro approccio sarà OOP perché crediamo che la nuova versione di PHP sia molto orientata a questo tipo di approccio. Perciò *$db* sarà un oggetto appartenente alla classe *SQLiteDatabase.* Il costruttore dell'oggetto inizializza il database che gli viene passato come parametro.

CREAZIONE DELLE TABELLE

Nella sintassi del codice nel **passo 2** è importante notare che abbiamo utilizzato le transazioni per creare la tabella e inserirvi dentro i dati. In particolare la transazione inizia con il costrutto *BEGIN*. A questo punto il database viene sottoposto a *"Lock"* e nessun'altra operazione in scrittura potrà essere eseguita fino al "Commit". Questo tipo di approccio ci mette al riparo da una serie di problemi. È sempre importante, ed a maggior ragione in SQLITE utilizzare le transazioni per scrivere dati nel database.
Un secondo elemento importante è che non esiste un metodo per determinare se una tabella esiste o meno prima di eseguire lo statement che la crea. Perciò se eseguite un refresh della pagina dopo avere già creato una prima volta la tabella, vi troverete davanti a un warning che vi avvisa che non potete creare una tabella che esiste già.
Per evitare questo problema è possibile utilizzare un trucco di questo genere:

```
if (! function_exists ('sqlite_table_exists')) {
  function sqlite_table_exists($db,$mytable) {
    $result = $db->singleQuery
    ("SELECT COUNT(*) FROM sqlite_master WHERE
                 type='table' AND name='$mytable'");
    return $result > 0;
  }
}
```

CREIAMO UN DATABASE

```
<?
$db = new SQLiteDatabase(
                         "ioProgrammo.db");
// creiamo il database. Nel crearlo dobbiamo
// ricordare che è importante che la
// directory che ospita il db deve avere i
// permessi in scrittura per l'utente che
// usa la pagina
?>
```

1 **Dopo avere eseguito questa istruzione vi troverete sul computer, il file *ioProgrammo.db* che conterrà la nostra base di dati.**

CREIAMO DELLE TABELLE

```
<? $db ->query("BEGIN;
  CREATE TABLE Redazione
  (id INTEGER primary KEY, name char(225));
  INSERT into Redazione (name)
                      values ('Fabio');
  INSERT into Redazione (cognome)
                      values ('Farnesi');
  INSERT into Redazione (name)
                      values (Raffaele);
  INSERT into Redazione (cognome)
                      values (Del Monaco);
  COMMIT; ");?>
```

2 **Questo spezzone di codice crea la tabella *Redazione*, che ha tre campi: *Id, Nome, Cognome* e due record contenenti i dati di due redattori.**

INTERROGHIAMO IL DATABASE

```
$result = $db->query
              ("select * from redazione");
while ($result->valid()) {
  $row = $result->current();
  echo "<b>$row[id])</b> $row[nome]
                 $row[cognome] <br>";
  $result -> next();
};
```

3 **Semplicemente eseguiamo una query e recuperiamo i dati tramite un ciclo di *While*. Si noti che stiamo usando un approccio ad oggetti.**

```
if (!sqlite_table_exists($db,"Redazione")) {
  $db ->query("BEGIN;
    CREATE TABLE Redazione (
    id INTEGER primary KEY, nome text, cognome text);
  INSERT into Redazione (nome,cognome)
                        values ('Fabio','Farnesi');
  INSERT into Redazione (nome,cognome)
                   values ('Raffaele','Del Monaco');
    COMMIT;
  ");
}
```

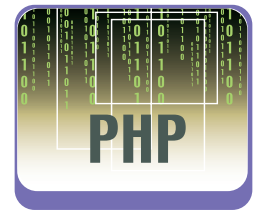
I database di tipo SQLite si portano in dotazione una tabella *sqlite_master*, al cui interno sono definite tutte le tabelle contenute nel database. La funzione *sqlite_table_exists* definita da noi non fa altro che interrogare *sqlite_master*, vedere se al suo interno c'è una entry corrispondente al nome di tabella passato come parametro, e restituire *true* o *false* a seconda del risultato trovato. Usare *sqlite_table_exists* prima della creazione di una tabella sarà infine un gioco da ragazzi.

QUERY DI INTERROGAZIONE

Non ci sono note importanti riguardo al **passo 3**. Si tratta di una normale query di interrogazione. Una nota potrebbe essere fatta rispetto all'uso degli iterator.
In sostanza, il codice che recupera i dati da un dataset potrebbe essere il seguente:

```
$result = $db->unbufferedQuery("select * from
                                  Redazione");
foreach ($result as $row)
{
  echo "<b>$row[id])</b> $row[nome]
                       $row[cognome] <br>";
}
```

Invece del normale metodo *'query'* abbiamo usato un metodo *'unbufferedQuery'*. Non ci sono grandi differenze con una query normale, eccetto che il risultato è una struttura sequenziale che può essere letta solo in una direzione una riga dopo l'altra. Non sono possibili accessi in punti predefiniti dell'array in quanto non ci sono Keys definite. Il vantaggio sta nel fatto che, essendo i cicli di foreach dei costrutti predefiniti del linguaggio, e non delle funzioni, dovrebbero essere leggermente più veloci di un normale fetch. Un esempio di iterator è contenuto nel **passo 5.**
Una considerazione analoga meritano il **passo 5** e il **passo 6**. Inizialmente abbiamo usato il metodo *singleQuery*, che restituisce il primo elemento di una colonna, in una stringa o in un array. Quindi abbastanza utile quando si prevede di recuperare un singolo dato.
Anche in questo caso il risultato è quello di ottimizzare l'uso delle risorse.
Nel **passo 6** invece abbiamo usato il metodo *arrayQuery* più un parametro *SQLITE_ASSOC*. Utilizzando questo metodo è facile coordinarsi con un *Template Engine* come *Smarty* ad esempio. Riporto qui solo i pezzi di codice senza addentrarmi nell'uso del template:

```
require('Smarty.class.php');
$smarty = new Smarty;
$result= $db->arrayQuery("SELECT * FROM
                    Redazione", SQLITE_ASSOC);
$smarty->assign('redazione',$result);
$smarty->display('index.tpl');
```

QUERY UNBUFFERED

```
$result = $db->unbufferedQuery(
              "select * from Redazione");
foreach ($result as $row) {
  echo "<b>$row[id])</b> $row[nome]
                  $row[cognome] <br>";
}
```

4 **Con il metodo *unbuffered* non si potrà accedere a dati casuali dell'array, ma si ha un'ottimizzazione dovuta all'uso del costrutto *foreach***

CONTIAMO I RECORD

```
<?
$result = $db->singleQuery("SELECT
COUNT(*) FROM Redazione");
echo "<br><b>Record inseriti</b>
$result"
?>
```

5 **Si noti che questa volta abbiamo usato il metodo *singleQuery*, che restituisce una stringa che contiene il valore della prima colonna di ogni riga.**

ARRAY ASSOCIATIVI

```
<?
require('Smarty.class.php');
$smarty = new Smarty;
$result= $db->arrayQuery("SELECT *
FROM Redazione", SQLITE_ASSOC);
$smarty->assign('redazione',$result);
$smarty->display('index.tpl');
?>
```

6 **Abbiamo usato il metodo *arrayQuery* unito al parametro *SQLITE_ASSOC* che ci restituisce un array associativo, utile se usiamo dei template.**

mentre il file *index.tpl* relativo potrebbe essere:

```
{section name=rdz loop=$redazione}
   <b>{$redazione[rdz].id})</b> {$redazione[rdz].nome}
                         {$redazione[rdz].cognome}<br>
{/section}
```

FUNZIONI PERSONALIZZATE

Una caratteristica piuttosto interessante di SQLITE è quella di potere definire delle funzioni personalizzate che possono essere utilizzate nelle query per il recupero dei dati. Consideriamo ad esempio la seguente funzione:

```
function cripta($str1,$str2)
{
            $str1.=$str2;
            return md5($str1);
}
```

IL TEMPLATE ENGINE SMARTY

In generale si indicano con il termine *Template Engine*" i sistemi che consentono di separare la logica della programmazione da quella di presentazione. In una programmazione in stile classico siamo abituati a considerare l'elaborazione l'output della pagina HTML come una parte della definizione del codice PHP. Questo ha lo svantaggio di impedire a un Web Designer di lavorare separatamente dal programmatore, con gravi danni psicologici per entrambi, e una sicura diminuizione della produttività. Entra perciò in gioco Smarty in quanto template engine. Considerate ad esempio

```
<?
 require('Smarty.class.php');
 $smarty = new Smarty;
 $contatore=0;
  while ($contatore < 10) {
     $ciclo[$contatore]=$contatore;
     $contatore++; };
  $smarty -> assign('ciclo',$ciclo);
  $smarty -> display('ioprogrammo.html');
?>
```

come vedete non c'e' l'ombra di HTML in questa sintassi. L'output di questa pagina viene demandato al file *ioprogrammo.html*. La definizione del file *ioprogrammo.html* potrebbe essere la seguente:

```
<html>
<title>ioProgrammo Esempio
Template</title>
<body>
{
   section name=rows loop=$ciclo
}
   <b>Ciclo:</b> {$ciclo[rows]} <br>
{/section}
</body>
</html>
```

Una chiara sintassi HTML con dei tag protetti dalle parentesi graffe tale che possa essere gestita anche da chi non ha conoscenza di programmazione PHP. Utilizzando questa tecnica programmatore e web designer possono lavorare in modo separato, con grande felicità di entrambi.

Si tratta molto semplicemente di una funzione che prende in input due stringhe, le concatena e restituisce una chiave criptata con algoritmo MD5. In sé e per sé non è certo una funzione complicatissima.
La useremo come esempio per mostrare quanto segue:

```
$db->createFunction('cripta','cripta',2);
$result = $db->Query("select nome, cognome,
                              cripta(nome,cognome)
                  as chiavesegreta  from Redazione");
```

Come vedete, abbiamo usato il metodo *createFunction* per creare una funzione utilizzabile dall'oggetto *$db* all'interno di una query. Intuirete che questo tipo di approccio in un certo qual modo ricalca quello classico della programmazione a oggetti. Infatti, supponiamo di avere utilizzato questa query centinaia di volte nel corso dello sviluppo del nostro software ed esserci resi conto che le chiavi restituite sono troppo semplici per implementare una reale sicurezza.
A questo punto tutto quello che dobbiamo fare è modificare la funzione *cripta($str1,str2)* e non tutte e 100 le query utilizzate all'interno del programma.

FUNZIONI DI UTILITÀ

Altra interessante caratteristica di SQLite è quella di inglobare alcune funzioni di utilità generale all'interno della classe *SQLiteDatabase*. Considerate ad esempio:

```
$numero_campi = $result->numFields();
  print $numero_campi;
  $counter='0';
  while ($counter < $numero_campi)
 {
      print $result->fieldName($counter++);
      print "<br>";
 }
```

vengono utilizzate due funzioni di utilità. La *numFields()* che restituisce il numero di campi contenuti nel risultato della query e la *fieldName()* che restituisce un array contenente i nomi di ciascun campo.
Abbiamo semplicemente inserito un ciclo per stampare il nome di ciascun campo.

CONCLUSIONI

È conveniente ed anche consigliato utilizzare SQLite in tutte quelle applicazioni dove non si prevede di dovere gestire la sicurezza a livello di singola tabella o singola colonna.
MySQL rimane comunque il più adatto in tutte le applicazioni che richiedono un Database più robusto e sicuro.

Un trucco ASP.NET per evitare sprechi di risorse

Pulsanti Intelligenti!

Vedremo come aggiungere del codice javascript a un pulsante per evitare richieste inutili al server, lo utilizzeremo in presenza di controlli di validazione e creeremo un controllo da inserire nelle pagine Asp.NET

Recentemente, mi sono imbattuto in un problema tanto facile apparentemente da risolvere quanto complesso nella realtà. Come evitare che una serie di continui click su un bottone, in un'applicazione ASP .NET, possa far decadere le prestazioni lato server? Le applicazioni web si basano su tecniche di PostBack, continui scambi di informazioni tra un browser e un server. Un doppio click su un bottone genera due chiamate al server. Nulla che un buon server non possa gestire, a meno che l'azione da eseguire in seguito ad un click non sia una transazione verso un database. In caso di transazioni complesse una continua pressione del pulsante potrebbe mandare in timeout il server.

UNA PRIMA SOLUZIONE

La prima idea è disabilitare il pulsante dopo che l'utente lo ha cliccato. Questo si può fare utilizzando la proprietà *Attributes* dell'oggetto *Button* e il relativo metodo *Add()*. Quest'ultimo accetta due parametri:

1. Il nome dell'attributo che verrà aggiunto al tag HTML.
2. Il nome della funzione o il codice client script che verrà eseguito dal pulsante.

Ecco un esempio:

```
myButton.Attributes.Add("onclick","this.disabled = true;");
```

Dopo aver chiamato la pagina, questa istruzione viene convertita in codice HTML:

```
<input type="submit" name="myButton" value="Ricerca"
id="myButton" onclick="this.disabled=true;" />
```

Premendo il pulsante sulla pagina web si può notare come il suo stato passa in disabilitato. Si nota anche che non viene più eseguito il postback verso il server! Quindi non viene più eseguito lo script lato server contenuto nell'evento *click* del pulsante.

ABILITIAMO IL POSTBACK

Per ovviare a questo malfunzionamento occorre aggiungere una chiamata al metodo *GetPostBackEventReference()* all'interno della classe *Page*. Questo metodo restituisce il codice javascript necessario a fare il submit della pagina, ovvero a contattare il server per fornirgli i dati contenuti nella pagina. Possiamo utilizzare i metodi e le proprietà di questa classe in quanto ogni pagina ASP.NET crea una classe derivandola da *Page*. Il codice diventa:

```
String strScript = "this.disabled = true;";
strScript += Page.GetPostBackEventReference(myButton);
myButton.Attributes.Add("onclick", strScript);
```

Nel metodo bisogna utilizzare un parametro per specificare il controllo al quale associare la chiamata postback verso il server. Ecco come verrà convertito il nuovo codice in HTML:

```
<input type="submit" name="myButton" value=
  "Ricerca" id="myButton" onclick="this.disabled=true;
                    __doPostBack('btnRicerca','');" />
```

Ovvero il codice completo delle funzioni di PostBack.

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste: ASP.NET e C#

Software: Visual Studio .NET 2003

Impegno

Tempo di realizzazione

OTTIMIZZAZIONI

Le tre semplici righe di codice che abbiamo scritto prima possono essere inserite, ad esempio, all'interno dell'evento *Load* della pagina:

```
private void Page_Load(object sender, System.EventArgs e)
{
String strScript = "this.disabled = true;";
strScript += Page.GetPostBackEventReference(myButton);
myButton.Attributes.Add("onclick", strScript); }
```

Così facendo riusciremo a disabilitare il pulsante per il tempo necessario a contattare il server per il submit dei dati e a ricevere la stessa pagina contenente il risultato dell'operazione richiesta come, ad esempio, un messaggio informativo sul buon esito di un'operazione verso il database. Ci sono, però, delle ottimizzazioni da fare per poter utilizzare questo codice all'interno di pagine dove esistono dei controlli di validazione tipo *RequiredFieldValidator*. Quando un controllo di tipo validazione viene inserito sulla web form, una serie di istruzioni lato client vengono aggiunte ad ogni controllo la cui proprietà *CausesValidation* che è impostata a *true*. Solitamente i pulsanti hanno questa proprietà attiva e quando sono renderizzati nella pagina HTML assumono questo codice:

```
<input type="submit" name="myButton" value="Ricerca"
    onclick="if (typeof( Page_ClientValidate) == 'func-
  tion') Page_ClientValidate(); " language="javascript"
                                        id="myButton" />
```

L'istruzione *typeof()* sostanzialmente controlla che la funzione *Page_ClientValidate* sia presente all'interno della pagina HTML e, in caso affermativo, la esegue. Senza apportare modifiche al nostro codice, il pulsante effettua il postback al server anche nel caso in cui la condizione controllata dal field validator non è soddisfatta. A questo punto non ci rimane che disabilitare la proprietà *CausesValidation* del pulsante impostandola a *false* e gestire manualmente la validazione del controllo utilizzando la seguente tecnica:

```
System.Text.StringBuilder sb =
                    new System.Text.StringBuilder();
sb.Append("if (typeof(Page_ClientValidate) ==
                                  'function') { ");
sb.Append("if (Page_ClientValidate() == false) {
                                      return false;
  } } ");
sb.Append("this.disabled = true;");
sb.Append(this.Page.GetPostBackEventReference(
```

NOTA

Una delle regole di programmazione proposta da Microsoft per le applicazioni Windows riguarda la visualizzazione del cursore a clessidra e la disabilitazione del pulsante dopo la sua pressione. Purtroppo però, almeno fino alla release 2.0 di ASP.NET, la disabilitazione del pulsante non è proprio di immediata realizzazione. Con questo articolo abbiamo visto come implementare una possibile soluzione.

COSA VUOL DIRE POSTBACK?

Le applicazioni ASP.NET sono basate su "PostBack". Nella programmazione Web tradizionale eravamo abituati a ragionare pensando ad una logica Form->Post->Action. L'idea era che una form posizionata in una pagina ASP passasse dei dati ad una seconda pagina tramite un POST generato dalla pressione di un pulsante. La seconda pagina prelevava poi i dati e li utilizzava per i propri scopi. In realtà la programmazione di applicazioni Internet non può prescindere da questo tipo di logica. ASP.NET, tuttavia fornisce all'utente un FrameWork avanzato che consente di astrarsi da queste conoscenze ed utilizzare direttamente controlli e programmazione ad eventi. In particolare, utilizzare un Bottone su una pagina ASP.NET genera il seguente codice nella pagina HTML:

```
<html>
<body>
<form name="myForm" method="post" action="test.aspx"
id="myForm">
<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE"
  value="dDwtMTAwOTA0ODU1NTs7PslICK1DzvZs
                              K0J6OQ2dxKqmEwVs" />
<script language="javascript">
<!--
 function __doPostBack(eventTarget, eventArgument) {
  var theform = document.myForm;
  theform.__EVENTTARGET.value = eventTarget;
  theform.__EVENTARGUMENT.value = eventArgument;
  theform.submit();
 }
// -->
</script>
<a id="Test" href="javascript:__doPostBack('Test','')">
                                   Create Text file</a>
</form>
</body>
</html>
```

Come si vede dalla riga

```
<a id="Test" href="javascript:__doPostBack('Test','')">
                                   Create Text file</a>
```

La pressione del pulsante richiama una funzione JavaScript che riceve due parametri

```
EVENTTARGET
EVENTARGUMENT
```

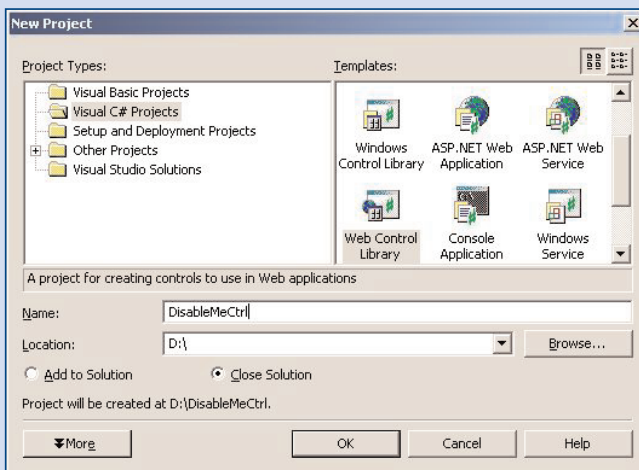
In cui *EVENTTARGET* contiene il nome del controllo che ha generato l'evento e *EVENTARGUMENT* contiene un eventuale argomento aggiuntivo. La funzione *__doPostBack(eventTarget, eventArgument)* non fa altro che inviare la *FORM* tramite un *POST* al server passando anche dei paramentri nascosti che serviranno ad individuare quali azioni devono essere intraprese lato server per gestire la pressione del pulsante.

```
                                    myButton));
sb.Append(";");
myButton.Attributes.Add("onclick", sb.ToString());
```

Questo semplice script concatena una stringa con le stesse istruzioni che vengono aggiunte automaticamente dalla proprietà *CausesValidation*, più le due istruzioni per disabilitare il pulsante e lanciare il submit verso il server. Quindi quando la condizione controllata dal controllo di validazione non è soddisfatta lo script esegue l'istruzione return false che evita di eseguire le successive istruzioni che effettuano la disabilitazione del pulsante e il postback verso il server.

Fabio Claudio Ferracchiati

CREAZIONE DI UN PROGETTO WEB CONTROL LIBRARY CON VISUAL STUDIO .NET

Per evitare di inserire lo stesso codice per tutti i pulsanti presenti su una o più pagine ASP.NET si può creare un server control derivato dalla classe *Button*. Tra l'altro questo metodo di programmazione, in caso di modifiche, permette di intervenire in un unico punto del codice e diffondere la modifica automaticamente tra le pagine.

1 **Dal menu *File* cliccare su *New* e poi scegliere la voce *Project*; all'interno della dialog box *New Project* scegliere il Visual C# Projects e il template *Web Control Library*. Inserire *DisableMeCtrl* come nome all'interno della casella di testo *Name* e premere *OK*.**

3 **Affinché il controllo sia utilizzabile in tutte le nostre applicazioni clicchiamo con il tasto destro sulla toolbox e selezioniamo "*add/Remove Items*" Selezionare l'etichetta .NET Framework Components, premere il pulsante *Browse...* e navigare nelle directory alla ricerca dell'assembly creato dalla compilazione precedente.**

```
[ToolboxData("<{0}:AutoDisable runat=server>
                                   </{0}:AutoDisable>")]
public class AutoDisable : System.Web.UI.WebControls.Button
{
protected override void Render(HtmlTextWriter writer)
  {
     System.Text.StringBuilder sb =
                          new System.Text.StringBuilder();
     sb.Append("if (typeof(Page_ClientValidate) == 'function') { ");
     sb.Append("if (Page_ClientValidate() == false)
                                   { return false; }} ");
     sb.Append("this.disabled = true;");
     sb.Append(this.Page.GetPostBackEventReference(this));
     sb.Append(";");
     writer.AddAttribute("onclick",sb.ToString());
     base.Render(writer);}
}
```

2 **Questo codice va inserito all'interno della pagina *WebCustomControl1.cs* che viene creata automaticamente dall'ambiente di sviluppo, sovrascrivendo la classe presente al suo interno. Compiliamo il codice utilizzando il menu *Build* e poi selezionando *Build Solution*.**

COME FUNZIONA

La prima istruzione serve per definire il tag HTML utilizzato dal controllo.
Successivamente viene creata una classe derivandola dalla classe *Button*. In questo modo si ottengono automaticamente tutte le caratteristiche di un controllo *Button* e, tramite il metodo virtuale *Render()*, si può definire una nuova caratteristica. Il parametro *writer* fornito da questo metodo, infatti, permette di scrivere codice HTML all'interno della pagina ASP.NET così da darci la possibilità di aggiungere il codice necessario alla disabilitazione del pulsante. L'ultima istruzione serve per richiamare il metodo *Render()* della classe padre, ovvero del *Button*. Questo è necessario per completare le istruzioni che trasformano il controllo *Button* con tutte le sue caratteristiche in codice HTML.
Fatto questo il nostro nuovo bottone dotato di funzionalità disable sarà pronto per essere usato in tutti i nostri progetti Web.

Messaggistica fra applicazioni aziendali: un nuovo modo

Achitetture basate su messaggi

Scopriremo Mantaray, un tool che può essere utile per creare sistemi distribuiti, faremo qualche esempio di automazione aziendale, e impareremo molto su Java Messaging Service

Ogni volta che una persona mi chiede cosa faccio per vivere, se sono ben disposto a chiacchierare, rispondo che lavoro nell'ambito dell'"Enterprise messaging business", e ogni volta ottengo la stessa risposta:

"È qualcosa che ha a che fare con MSN Messenger o AIM?"

La risposta è "No" direi di "No".

"Messaging" è un termine dai molteplici significati nel campo dell'informatica. Ci si può riferire, con esso, a qualsiasi cosa dai piccioni viaggiatori alle Email. "Enterprise Messaging", tuttavia è un termine usato per descrivere la comunicazione asincrona fra applicazioni di tipo aziendale. In questa accezione, un messaggio è da intendersi come un contenitore di dati formattati. Tali dati vengono tipicamente generati da un'applicazione di tipo Business. Ogni messaggio è concepito in maniera tale che possa essere letto da altre applicazioni; queste che ricevono il messaggio, compiono delle operazioni basate sul significato del suo contenuto. Perciò non è sempre vero che un messaggio nasca con lo scopo di essere fruibile per un essere umano. Da qui in poi, ogni volta che useremo il termine "messaggio", si dovrà intendere nell'accezione dell'*enterprise messaging*, ovvero quello di cui parleremo in questo articolo.

L'IDEA

La chiave per il buon funzionamento di un'applicazione di automazione aziendale è tipicamente costituita dal numero di messaggi che i partecipanti a un processo di produzione riescono a scambiarsi e dalla qualità dei messaggi stessi.
Cosa succederebbe se non ci fosse bisogno di un intervento umano nello scambiarsi i messaggi?
Ad esempio, il mio PC potrebbe comunicare ad un altro PC che è acceso e che sta lavorando, perciò sono in ufficio. Il PC che riceve il messaggio potrebbe aggiornare un suo database contenente informazioni amministrative e poi inviare un secondo messaggio a un PC che si trova nel reparto Grafica. Il PC che lo riceve potrebbe comunicarmi con un ulteriore messaggio che un certo articolo che aspettavo per l'impaginazione è pronto e possiamo passare alla revisione.
Come vedete, un sistema di messaggistica aziendale, dove per messagistica si intende la capacità delle macchine di scambiarsi informazioni, cambia completamente il concetto di sviluppo di Intranet.

I SISTEMI DI MESSAGGISTICA

Ci sono molte tipologie di sistemi di messaggistica, basati su logiche diverse. Quello che tutti hanno in comune è la capacità di inviare e ricevere messaggi. Ciascun client (un'applicazione che usa Mantaray per esempio) connesso a un sistema ha la capacità di creare un messaggio, riempirlo con dei dati e mandarlo a ogni altro client che utilizzi lo stesso sistema. Alcuni di essi permettono il broadcasting di un messaggio verso un certo numero di applicazioni in ascolto, altri consentono di inviare un messaggio solo a una singola destinazione. Alcuni sistemi si basano su un'interfaccia asincrona, tale che il messaggio sia ricevuto dal client non appena esso viene spedito. Altri sistemi usano un'interfaccia sincrona tale che il client debba richiedere il messaggio per poterlo ricevere. Un sistema di messaggistica può inoltre implementare un certo numero di funzioni aggiuntive. Per esempio, la persistenza dei messaggi, certezza della consegna/ricezione, policy di autenticazione, priorità dei messaggi, messaggi time-to-live e altri.

COS'È JMS ?

JMS, Java Message Service, è un insieme di interfacce tale che un client basato su Java possa fare uso di un sistema di messaggistica. Definisce insomma uno standard e fornisce delle funzioni per accedere

allo standard. Fornisce un ricco ma semplice insieme di funzionalità, tali da consentire la creazione di applicazioni enterprise molto potenti. Un JMS Provider è un entità che fa uso di JMS. L'insieme di funzioni esistenti per JMS è abbastanza grande da consentire di interagire con sistemi di messaggistica diversi. JMS definisce due tipologie di messaggi: *peer-to-peer (PTP)* e *publish-and-subscribe (Pub/Sub)*. Lo scambio di messaggi PTP si basa sul concetto di coda *(queue)*. I client possono mandare messaggi ad una coda specifica e altri client possono accedere alla coda e consumare i messaggi da essa. Questo sistema si utilizza prevalentemente nella costruzione di applicazioni basate su uno scambio di messaggi in modo sincrono. Il sistema *Pub/Sub* introduce il concetto di argomento *(topic)*. Un client si iscrive (effettua un subscribe) ad un topic per poter ricevere messaggi riguardanti uno specifico argomento. Gli altri client pubblicano *(publish)* i messaggi verso questo topic e i messaggi vengono spediti a tutti i client che sono iscritti al topic in questione.

MANTARAY E JMS

MantaRay (*www.MantaMQ.org*) è un sistema di messaggistica opensource, che ha caratteristiche tali da renderlo idoneo in sistemi server-less, ovvero privi di un server centrale, perciò basati su un'architettura puramente distribuita. La maggior parte dei sistemi di messaggistica utilizza un broker centralizzato o un insieme di brokers. Utilizza cioè un server centrale che faccia da mediatore per lo smistamento dei messaggi. Ciascun client si connette a un broker, che funge da gateway verso il domino di messaggistica. Il broker mantiene traccia di tutti i differenti client, oltre ovviamente a gestire le sottoscrizioni a un topic e a provvedere al mantenimento delle varie code. Ciascun messaggio inviato al broker, viene immagazzinato fino a che un client non lo richiede (nel caso di messaggi di tipo PTP) o spedito ai vari client sottoscritti a un topic (nel caso di messaggi di tipo PTP). Mantaray utilizza esattamente lo stesso metodo di tutti gli alri sistemi, eccetto che per un particolare molto importante, non c'è necessità di nessun broker. Facendo uno sforzo di immaginazione si può pensare che ciascun client funzioni come una sorta di mini-broker. Ciascun client è a conoscenza di ogni altro client all'interno del sistema (di qualunque tipo esso sia, queue producer, consumer, un elemento che ha sottoscritto un'iscrizione a un topic). Ogni volta che un client ha bisogno di mandare un messaggio, è gia dotato di tutte le informazioni di cui ha bisogno per mandarlo direttamente a tutti gli altri client presenti sulla rete. Diamo uno sguardo a come questo viene realizzato

Automatic discovery

Con un'architettura basata su broker tutte le applicazioni presenti nel network, "conoscono" soltanto il broker, mentre il broker "conosce" tutte le applicazioni. Quando l'applicazione *A* vuole inviare un messaggio all'applicazione *B* lo manda al broker e il broker lo manda all'applicazione *B* e viceversa.
Poiché MantaRay è un sistema distribuito e non ha bisogno di un broker, gli elementi in una rete che fa uso di Mantaray hanno bisogno di conoscersi a vicenda in modo diretto. Per raggiungere questo scopo, il sistema avvia un processo di "automatic discovery" usando un canale multicast. Tutte le informazioni come IP, porte e altre informazioni (ad esempio se l'applicazione A è un subscriber per un certo topic) sono pubblicate in un unico repository multicast. Tutti gli elementi di Mantaray ascoltano su un canale per ricevere informazioni sui loro peers. Dopo che l'applicazione *A* ha scoperto l'applicazione *B* essa è abilitata ad aprire una connessione TCP/SSL di tipo Peer-To-Peer.

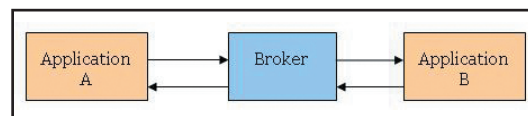



Fig. 1: ***In un sistema basato su Broker centralizzati, è il broker ad occuparsi di smistare i messaggi***

Topics

Le applicazioni possono comunicare l'una con l'altra usando I topic: una o più applicazioni sottoscrivono un topic e le altre applicazioni pubblicano I dati nel topic. Ci possono essere un numero qualunque di iscritti e di publisher per ciascun topic nel sistema. Un messaggio pubblicato da un publisher sarà ricevuto da tutti gli iscritti; un iscritto potrebbe anche essere offline (ad esempio il processo è down) e ricevere comunque il messaggio pubblicato quando diventa di nuovo attivo. Con un'architettura basata su broker, i messaggi e le sottoscrizioni sono gestite da un broker centralizzato. Tutti i messaggi pubblicati vengono inviati al broker; e questo è responsabile dell'invio del messaggio a tutti gli iscritti. Mantaray aggira il concetto di broker di modo che tutte le comunicazioni vengono effettuate in modalità peer-to-peer. I publisher possono mandare i messaggi direttamente agli iscritti. Le comunicazioni sono basate su TCP e possono essere criptate usando SSL.

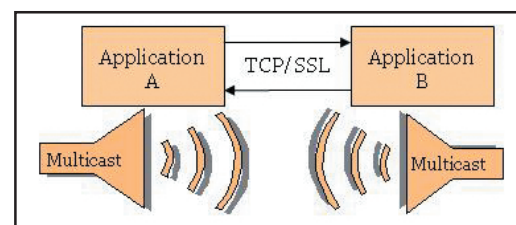



Fig. 2: ***Mantaray usa un canale multicast per annunciare la presenza di un host alla rete***

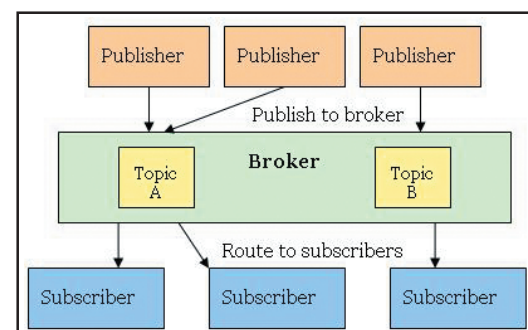



Fig. 3: ***In un sistema basato su broker è il broker a gestire i topik***

Queues
Le applicazioni possono comunicare fra di loro usando code di messaggi. Un'applicazione invia messaggi ala coda e un'altra riceve i messaggi dalla coda. Ci possono essere un numero qualunque di message producers e di message consumers su una data coda e ci possono essere un numero illimitato di code nel sistema. I messaggi vengono mantenuti nella coda nell'ordine in cui sono stati inviati fino a che un consumer non richiede un messaggio. Ciascun messaggio sarà inviato a uno e un solo consumer, poi verrà eliminato dalla coda. Questo tipo di architettura rappresenta la differenza principale fra le code e i topics. I consumer possono registrarsi come "listeners" su una coda e nel fare questo eliminano la necessità di richiedere il messaggio ogni volta. Nella modalità *"queue listeners"* ogni messaggio ricevuto dalla coda sarà inviato immediatamente ai suoi listeners. Con un'architettura basata su broker tutte le code sono gestite da un broker centralizzato in maniera molto simile a un topic. In Mantaray è stato introdotto anche un terzo ruolo (diverso da consumer o producer) chiamato queue coordinator, il quale può risiedere in ogni elemento di tipo MantaRay all'interno del network. Differenti tipologie di elementi Mantaray possono coordinare code differenti. La sola regola che deve essere mantenuta è che c'è solo un coordinatore per coda. Il coordinatore di ciascuna coda è un attributo configurabile di MantaRay. Una volta che un elemento è stato configurato come coordinatore di una coda, automaticamente inizierà a gestirla. Il coordinatore di una coda può anche essere un oggetto *MantaRay* dedicato che non agisce come consumer o producer completamente.

Fig. 4: In un sistema basato su Mantaray l'architettura è completamente distribuita

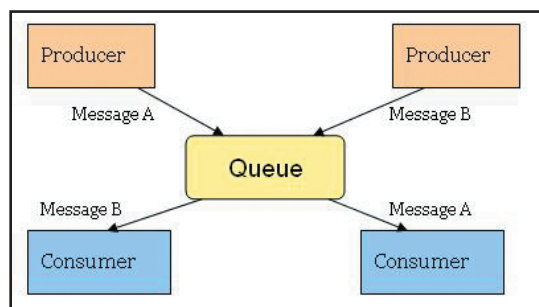



Fig. 5: I producer inviano alla coda e i consumer ricevono dalla coda

NESSUN PUNTO DI CONGESTIONE

Un sistema di messaggistica sufficientemente grande, gestisce un numero enorme di messaggi al minuto. Nel caso di architetture centralizzate, tutti questi messaggi vengono processati da un broker centralizzato. Questo richiede l'uso di un server abbastanza potente con un processore ed una quantità di memoria adeguati a gestire questa grande massa di messaggi. Per la maggior parte dei sistemi questa rimane un'asserzione del tutto teorica; infatti solitamente un solo server non è sufficiente, è necessario effettuare il load balancing, avere un server di backup e un gruppo di continuità adeguato a gestire le assenze di energie sulla totalità dei server. Molti soldi devono essere spesi per costruire un'infrastruttura di rete che possa gestire il fatto che tutti i messaggi devono necessariamente passare attraverso un'unica struttura.

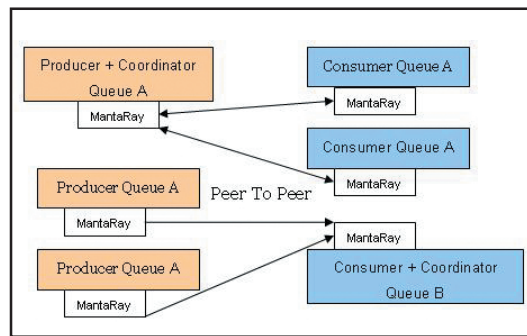



Fig. 6: Uno schema di funzionamento di Mantaray

Con un sistema distribuito, cade l'esigenza di avere un server centrale dotato dei requisiti hardware adeguati. Infine,; poiché i messaggi vanno direttamente da client a client saltando i punti intermedi diminuisce chiaramente il consumo di banda all'interno del network aziendale. In più i sistemi distribuiti sono davvero molto scalabili. Per esempio, se il numero di messaggi improvvisamente cresce in corrispondenza dell'introduzione di una nuova applicazione o in corrispondenza all'arrivo di molti nuovi computer non c'e' bisogno di comprare un nuovo server e l'aumento del consumo di banda e distribuito in modo omogeneo.

NESSUN PUNTO DI DEBOLEZZA

Per natura un sistema centralizzato è denominato tale perché al suo interno ci sarà un componente senza cui l'intero sistema cessa di funzionare. Nonostante sia sempre possibile alzare il livello di affidabilità a mezzo dell'introduzione di linee di backup, il punto è che c'è sempre una possibilità anche piccola di essere soggetti a downtime. Se si evita di centralizzare il sistema e anche di avere un singolo punto di debolezza all'interno del sistema.
Certamente i client possono ancora andare soggetti a malfunzionamenti, e peggio ancora, può essere il queue coordinator ad avere un problema, per cui una specifica coda potrebbe cessare di funzionare. Ma progettando bene il sistema, ad esempio evitan-

do che ci sia un singolo client che coordina tutte le code, elimineremo la possibilità che un guasto interrompa il funzionamento dell'intero sistema.

SEMPLICE DA INSTALLARE

Poniamo il caso che lavori in un piccolo team o anche in una grande azienda. In tutti e due i casi con personale limitato e un reparto tecnico subissato di lavoro. Se volete iniziare a utilizzare un sistema di messaggistica per automatizzare i processi di lavoro, avete due opzioni. Potreste aspettare 3 mesi fino a che arrivi un nuovo computer su cui installare un broker centrale, e poi una settimana o due prima che il reparto tecnico trovi il tempo di installare un complicato sistema che è sufficientemente vicino a quello di cui hai bisogno; oppure puoi scaricare un semplice sistema di messaggistica distribuita, installare un client su ciascun computer e potrai vedere i messaggi iniziare a volare da un posto all'altro in un tempo assolutamente minimo!

JMS E PHP POSSONO INTERAGIRE

PHP è un linguaggio di scripting molto potente che è popolare per essere usato in congiunzione ad Apache. Benché si stia sforzando di diventare un linguaggio di scripting di back-end, tuttavia è ancora molto limitato all'essere un linguaggio per applicazioni di front-end. Il punto è che si possono creare applicazioni molto interessanti combinando l'uso di Java e PHP. PHP può essere usato per la sua versatilità nella costruzione di Web Application e per la capacità di coordinarsi facilmente con molti tipi di database. Come strumento di back-end utilizzeremo la forza delle API di Java Messaging. PHPMQ è un progetto opensource che fornisce un toolkit per il messaging a PHP. Aggiunge a PHP la possibilità di di eseguire operazioni JMS come il mandare e ricevere messaggi sulle code e sui topic. Il repository originale di PHPMQ è disponibile presso *http://sourceforge.net/projects/phpmq/*. In futuro pensiamo di creare delle interfacce anche per Perl e per Python, o magari un web services per consentire ai linguaggi non-Java di interagire con JMS

Fig. 7: ***PHPMQ consente di interfacciare PHP con Mantaray***

UN PROGRAMMA D'ESEMPIO

Noi abbiamo un call center per il supporto. Un operatore risponde alle chiamate e apre una richiesta di supporto se necessario, l'helpdesk riceve la richiesta di supporto e chiama il cliente per aiutarlo. L'operatore ha un'applicazione grafica scritta in Java (*swing*), il layer di presentazione di questa applicazione chiama un oggetto chiamato *RequestDispat-*

PER INIZIARE

È necessario scaricare Mantaray da http://sourceforge.net/projects/mantaray. Oppure prelevarlo direttamente dal CD di ioProgrammo. Inizieremo scompattando il file contenente i binary in una qualche directory sull'hard disk. Ci riferiremo a questa directory con il nome di Manta Home. Ciascun client MantaRay dovrebbe girare in una propria directory. Questo significa che se volete fare una prova facendo girare due client su uno stesso computer, dovreste utilizzare due directory di installazione separata. Per la nostra prova utilizzeremo due computer.
Editiamo il file *default_config.params* come segue:

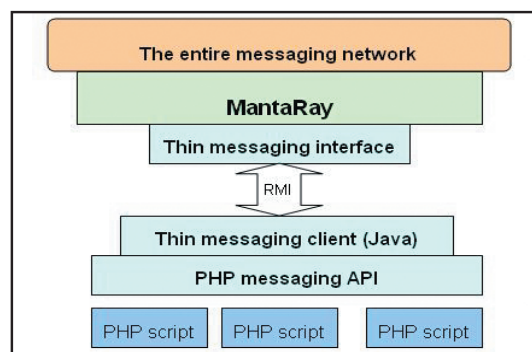
```
coordinated_queues = Q1
net.this.agent.name=Q1Agent
net.this.agent.defaultDomain=myDomain
rmi_registry_port = 10005
```

e il file *world.xml* come segue

```
<world ver='5' >
<domain name='myDomain' desc='This is the predefined default domain.' >
<service name='Q1' serviceType='queue' persistent='false' />
<agent name='Q1Agent' >
<transport ip='172.16.0.25' port='6667' type='TCP' />
</agent>
</domain>
</world>
```

Sostituite i valori degli IP con quelli desiderati. Configurando i due file in questo modo, avremo creato una coda di nome Q1, coordinata da un agent chiamato Q1Agent. Allo stesso modo sul secondo computer avremo creato una coda Q2 coordinata da Q2Agent. A questo punto sui due computer facciamo partire il file standalone.bat. Partirà l'autodiscovery e i due mantaray si saranno riconosciuti. Possiamo stoppare standalone.bat. Siamo pronti per fare partire la nostra prima applicazione. Useremo uno degli esempi presenti nel pacchetto. Perciò dalla home di mantaray possiamo digitare:

```
java -DmantaHome="c:/mantaray" -cp c:/talk;c:/mantaray/manta.jar;c:/
    mantaray/ext/jms.jar;c:/mantaray/ext/log4j-
    1.2.8.jar;c:/mantaray/ext/jgroups.jar;c:/mantaray/ext/antlr.jar
    sample.jms.queues.Talk.Talk -u SALES -qr q1 -qs q2
```

avviate sull'altro computer un client con la stessa riga di comando ma con le code di ricezione e invio invertite, et voilà il gioco è fatto.

NOTA

SUPPORTO

Il team di MantaRay è sempre felice di supportare gli utenti MantaRay, Potete postare una richiesta nel forum o nella mailing list per ogni problemi. Siamo anche molto contenti di ricevere richieste su nuove funzionalità da aggiungere a MantaRay, oppure segnalazioni di Bug, report e contributi al codice. Tutte queste informazioni sono reperibili presso http://sourceforge.net/projects/mantaray/

GLI AUTORI

Amir Shevat è uno sviluppatore senior con otto anni di esperienza nel settore. È un esperto di sviluppo Java ed ha una vasta esperienza in C++, Symbian e PHP. È il responsabile del team di sviluppo dei progetti Open Source di Coridan Inc, Mantaray e PHPMQ.

Uri Shneider è uno sviluppatore senior con oltre 12 anni di esperienza. Le sue conoscenze vannno da TCP/IP alla programmazione in ambiente linux. Solitamente preferisce le piccole compagnie in fase di start-up e al momento lavora in Coridan Inc, nel team di svilyppo di Mantaray.

cher che invia una richiesta d'aiuto all'helpdesk. Il *RqustDispatcher* usa le API JMS di Mantaray per inviare il messaggio alla coda corretta. Il metodo *RqustDispatcher* potrebbe assomigliare a qualcosa del genere:

```
…
public void dispatchRequeust(String
        supportRequestDetails ) throws Exception{
  QueueConnectionFactory conFactory =
        (QueueConnectionFactory) new
    MantaQueueConnectionFactory();
  javax.jms.QueueConnection con =
        conFactory.createQueueConnection();
  // This session is not transacted
  javax.jms.QueueSession sendSession =
        (QueueSession) con.createQueueSession(false,
    Session.AUTO_ACKNOWLEDGE);
  javax.jms.Queue sendQueue =
    sendSession.createQueue ("SUPPORT_REQUEST");
  javax.jms.QueueSender sender =
        sendSession.createSender(sendQueue);
  // Send the support request to the queue
  javax.jms.TextMessage msg =
        sendSession.createTextMessage();
  msg.setText(supportRequestDetails);
  sender.send(msg, javax.jms.DeliveryMode.NON_
  PERSISTENT, javax.jms.Message.DEFAULT_PRIORITY,
        MESSAGE_TTL);
}
```

IL LATO PHP

Lo staff dell'Helpdesk usa un'applicazione PHP per ricevere le richieste di supporto (quelle che sono state messe in coda dall'operatore) e agisce su queste.
Lo script PHP potrebbe assomigliare a qualcosa del genere:

```
<html>
<body>
…
<?php
include('./libraries/messaging.php');
// first parameter is the host name of the
// MantaRay RMI registry(local is localhost),
// second parameter is the port the RMI registry
// is listening to
$msgAPI = new messaging('localhost',10005);
// the first parameter in the dequeue is a unique
// key of the PHP MantaRay session. The second
// parameter is the name of the queue.
$supportRequest = $msgAPI->dequeue('helpDesk', '
        SUPPORT_REQUEST');
if($supportRequest == null){
  supportRequest = "No support request, go to lunch J");
}
?>
<h1> Help desk center </h1>
<br/>
<h2> Next support request </h2>
<br/>
<h3>
<?php echo &supportRequest?>
</h3>
<br/>
For the next support request in queue refresh page by..
…
</body>
</html>
```

CONFIGURARE MANTARAY

Diamo uno sguardo ai file presenti nella directory config/ nella Manta Home:

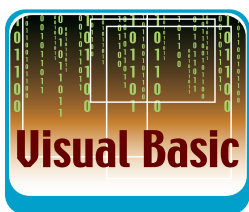
- **default_config.params** – questo file contiene i vari parametri di configurazione di Mantaray, con una spiegazione per ciascun parametro.
- **component_config.params** – questo file sovrascrive i parametri presenti in default_config .params. Qui dovreste inserire il layer name di mantaray e il suo dominio, oltre a tutte le informazioni diverse da quelle di default.
- **world.xml** – questo file definisce la World Map, che contiene i seguenti dati
  - **Domain name** – dovrebbe essere lo stesso valore settato in component_config.params.
  - **Service information** – quail topic e code sono disponibili nell sistema
  - **Transport information** – Per ciascun layer di trasporto, deve essere specificata una entry nel file *world.xml*
- **jgroups.xml** – contiene le definizioni dei *JGroups*, un toolkit multicast, usato per l'auto discovery. È importante che le informazioni in questo file siano le stesse per ogni installazione di Mantaray.
- **jg-magic-map.xml** – Server per JGroups. Non c'è bisogno di modificarlo
- **logger.config** – viene usato per definire il livello di logging del sistema, il nome del file di log e le altre cose relative alla gestione dei log.

Questo è tutto, siamo pronti al via.

Amir Shevat e Uri Shneider

Un database a basso costo ma incredibilmente potente

Usare MySQL in .NET con C#

Ecco come sfruttare interamente la potenza di MySQL con la facilità di sviluppo degli strumenti Visual Studio di Microsoft. Il tutto con un occhio di riguardo ai costi

Utilizza questo spazio per le tue annotazioni

Lo sviluppatore .NET è generalmente abituato ad accedere a dati presenti su basi dati Microsoft, come MSSQL Server o MSDE, la versione leggera di SQL Server distribuita con .NET. Spesso sviluppa anche in Oracle. Il problema è che, in versione completa, questi sistemi sono molto costosi e non alla portata di progetti dotati di basso budget. Sebbene siano sempre di più i fornitori di hosting che permettono l'utilizzo di SQLServer, la quasi totalità di questi operatori fornisce MySQL come database preferenziale per le applicazioni.
Il modello di licenza e la stretta interoperabiltà con PHP, ha permesso una grande diffusione si MySQL fra i fornitori di hosting web.

E .NET?

Il rilascio di molti provider managed, in primis quello scaricabile dal sito MySQL, il .NET Connector, insieme alla realizzazione di strumenti di gestione e interrogazione più amichevoli, hanno aperto il mondo MySQL anche agli sviluppatori .NET.
In questo articolo utilizzeremo il provider .NET fornito da MySQL. È possibile anche utilizzare altri provider managed che è possibile reperire in internet, oppure utilizzare il driver ODBC, il *MySQL ODBC-Connector*, nello stesso modo in cui utilizziamo tutti i fornitori di dati *ODBC*. L'utilizzo di un provider .NET però ha molti vantaggi, come le migliori prestazioni ottenibili e l'uso delle stesse tecniche cui lo sviluppatore .NET è abituato per l'accesso ai dati tramite i provider forniti con .NET.

COMINCIAMO!

Una volta installato il server e il provider e creato il nostro primo database come descritto nelle due procedure guidate che corredano questo articolo, possiamo cominciare a vedere come interagire con MySQL dalla nostra applicazione .NET. Per prima cosa dobbiamo inserire fra i riferimenti del progetto il provider mySQL. Per far questo, clicchiamo col pulsante destro sull'elenco di references, e scegliamo *Add References*. Possiamo quindi sfogliare il disco alla ricerca del provider. Generalmente questo viene installato nella cartella *C:\Programmi\ MySQL Connector-Net\bin\.NET 1.1.* A questo punto possiamo utilizzare il namespace adatto:

```
using MySql.Data.MySqlClient ;
```

Proviamo una prima, semplice query: prima apriamo una connessione:

```
MySqlConnection MyConn= new MySqlConnection(
    "server=localhost; database=marco1; user id=root;
                                       password=pwd;");
MyConn.Open();
```

poi creiamo un *MySqlCommand*, che eseguirà la query:

```
MySqlCommand Command= new
  MySqlCommand("SELECT * FROM utente ",MyConn );
```

per avere i risultati, chiamiamo il metodo *ExecuteReader*, che andrà a inizializzare un *MySqlDataReader:*

```
MySqlDataReader DR= Command.ExecuteReader(
          System.Data.CommandBehavior.SingleResult);
```

a questo punto possiamo ciclare nei risultati ottenuti. Nel nostro esempio scriviamo in debug i risultati, poi chiudiamo la connessione:

```
while (DR.Read())
{ System.Diagnostics.Debug.WriteLine(
            DR[0].ToString()+", "+DR[1].ToString()); };
MyConn.Close();
```

QUERY CON PARAMETRI

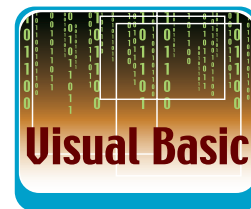
Possiamo utilizzare anche i parametri nelle nostre query. Vi consiglio di utilizzare sempre i parametri nelle query e di non usare mai la concatenazione di stringhe. Il meccanismo di creazione dei parametri infatti si prenderà cura di fare tutte le dovute sostituzioni e pulizie nei dati: banale esempio è la sostituzione del singolo apice con il doppio apice.
La query dell'esempio precedente può quindi essere sostituita dalla seguente:

```
MySqlCommand Command= new MySqlCommand(
  "SELECT * FROM utente where login=?p1",MyConn );
Command.Parameters.Add("p1","uno");
MySqlDataReader DR= Command.ExecuteReader(
         System.Data.CommandBehavior.SingleResult);
```

in questo modo verrà restituito il solo risultato voluto. Si noti come, a differenza di ciò che avviene per altri sistemi, come ad esempio SQL Server, per cui i parametri sono individuati dal carattere '@', in MySQL utilizziamo '?'.

SEMPLIFICHIAMO...

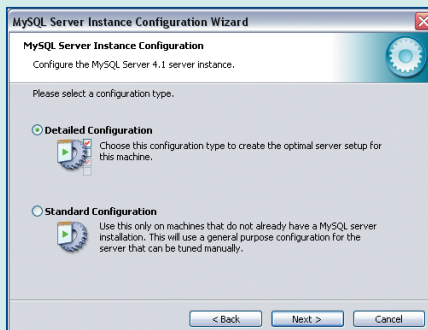
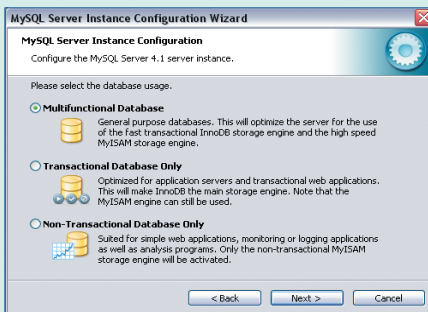
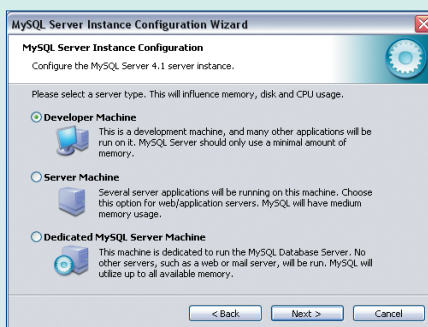
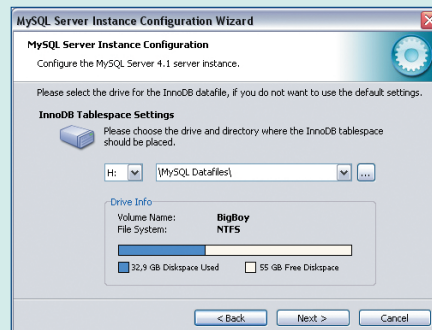
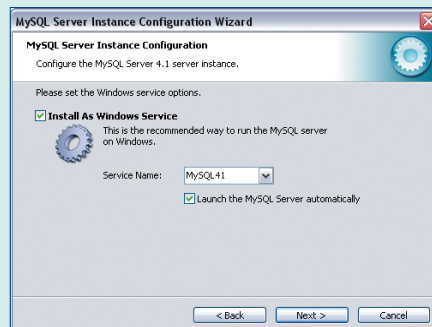
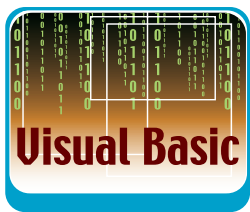
Qualora dovessimo avere un risultato scalare, ovvero un singolo risultato e non una o più righe, possiamo utilizzare il metodo ExcuteScalar, come nel prossimo esempio:

```
MyConn.Open();
MySqlCommand Command1= new MySqlCommand(
```

INSTALLARE MYSQL

Ecco i passi che dobbiamo seguire per installare MySQL nel nostro sistema e cominciare ad utilizzarlo da .NET

1 DOWNLOAD DEI PACCHETTI NECESSARI - Per prima cosa dobbiamo scaricare i pacchetti necessari da http://dev.mysql .com.
I due pacchetti coinvolti sono MySQL 4.1 e il .NET Connector.

2 INSTALLAZIONE DEL DBMS - l'Installer di Windows ci guiderà nei vari passi della configurazione. Se vogliamo personalizzare l'installazione, possiamo scegliere la custom, altrimenti la typical.

3 CONFIGURATION WIZARD - Durante l'installazione ci verrà chiesto se vogliamo configurare subito il server. Rispondere Si significa che il wizard ci assisterà nella prima configurazione.

4 STANDARD O DETTAGLIATA - Possiamo scegliere se intendiamo operare una configurazione dettagliata. Scegliamo questa opzione.

5 TIPO DI INSTALLAZIONE E TIPO DB - Per sviluppare software la developer va benissimo. Lasciamo le impostazioni di default se non abbiamo particolari esigenze.

6 SCELTA DELLA POSIZIONE DEI FILES - Scegliamo un disco dove abbiamo lo spazio necessario. Per il resto possiamo lasciare le impostazioni di default.

7 INSTALLAZIONE COME SERVIZIO E UTENTE - Ci resta solo da specificare se vogliamo che MySQL sia un servizio, assegnargli il nome e creare un utente amministrativo.

NOTA

MySQL è il più diffuso *DBMS (Data Base Management Server)* Open source al mondo. È un potente DBMS relazionale, scaricabile e utilizzabile liberamente secondo le licenze tipiche del mondo OpenSource. Negli anni MySQL è diventato un sistema complesso e potente, affidabile al punto da essere usato da importanti multinazionali e pubbliche amministrazioni. È possibile scaricare liberamente dal sito www.mysql.com il database server col nuovo installer, gli strumenti GUI di amministrazione e interrogazione, i driver ODBC ma soprattutto il provider managed nativo per .NET ne fanno un'opportunità davvero imperdibile anche per lo sviluppatore .NET.

```
        "SELECT count(*) FROM  utente  ",MyConn );
int i= (int)Command1.ExecuteScalar ();
MessageBox.Show("il sistema ha "+i+" utenti");
MyConn.Close();
```

A volte è necessario eseguire una query che non ritorna risultati. Esempi sono le query DDL, o le *INSERT* e *UPDATE*. In tal caso possiamo utilizzare il metodo *ExecuteNonQuery*, che ritorna soltanto il numero di righe modificate dal comando. Nel seguente esempio utilizzeremo questa possibilità per inserire una riga nella tabella:

```
MyConn.Open();
MySqlCommand Command2= new MySqlCommand(
     "insert into marco1.utente values (?login,?email,
                                    ?password)",MyConn );
Command2.Parameters.Add("login","marco");
Command2.Parameters.Add("email","email@marco.mmm");
Command2.Parameters.Add("password","pwd");
int r=Command2.ExecuteNonQuery();
MessageBox.Show("il sistema ha inserito "+r+" righe");
MyConn.Close();
```

DATI DISCONNESSI

Come avviene per il provider di SQL Server, anche il provider MySQL fornisce la classe *DataAdapter:* il *MySqlDataAdapter.* Il *DataAdapter* è un oggetto che permette di automatizzare le comuni operazioni di selezione, inserimento e aggiornamento dei dati disconnessi. Viene spesso utilizzato in congiunzione con le classi contenitrici di dati disconnessi, ovvero il *DataSet*, la *DataTable* e la *DataRow.* Il metodo utilizzato per riempire una *DataTable* è *Fill.* Nel prossimo esempio infatti creiamo un *DataAdapter* a partire da una query e una connessione:

```
MySqlDataAdapter DA= new MySqlDataAdapter(
                          "select * from utente",MyConn);
```

creiamo la *DataTable* che ospiterà i dati ricevuti:

```
DataTable DT= new DataTable();
```

e "riempiamo" i la *DataTable* con i dati:

```
DA.Fill(DT);
```

possiamo visualizzare facilmente questi dati, per esempio facendo il *DataBinding* su una *dataGrid* che avremo in precedenza posto nella nostra applicazione:

```
dataGrid1.DataSource=DT;
```

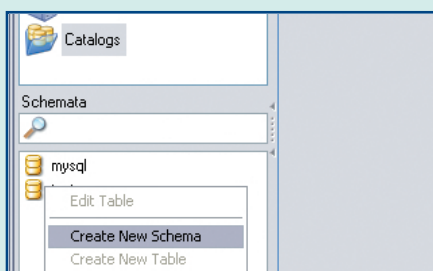
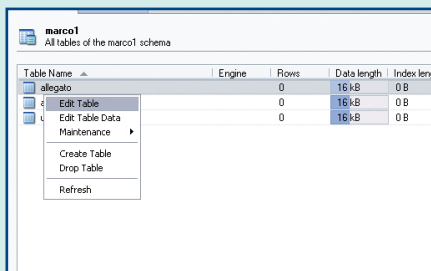
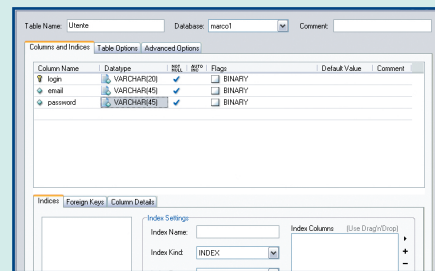
L'operazione di *DataBiding* può essere effettuata anche con sul *DataReader,* ma solo per le *DataGrid Web.* Qualora volessimo immettere attraverso una sola query dati in più tabelle disconnesse, potremmo utilizzare il *DataSet.*

```
DataSet DS= new DataSet();
MySqlDataAdapter DA1= new MySqlDataAdapter(
  "select * from utente; select * from articolo",MyConn);
DA1.Fill(DS);
```

Occorrerebbe però utilizzare il *DataMapping* per far sì che il *DataAdapter* riempia con i dati le *DataTable* con i nomi che vogliamo: altrimenti i dati verranno inseriti in *Table, Table1, Table2* e così via. Nel caso di un *DataSet* semplice, con due tabelle:

```
DataSet DS= new DataSet();
MySqlDataAdapter DA1= new MySqlDataAdapter(
  "select * from utente; select * from articolo",MyConn);
DA1.TableMappings.Add("Table","Utente");
DA1.TableMappings.Add("Table1","Articolo");
DA1.Fill(DS);
```

Così facendo possiamo trovare le tabelle "per nome":

CREARE UN DB IN MYSQL

1 CREARE UNO SCHEMA - Apriamo *MySQL Administrator*, selezioniamo *Catalogs* e facciamo clic col pulsante destro sul pannello inferiore. *AddNewSchema* e il gioco è fatto.

2 CREARE UNA TABELLA - Aggiungiamo una tabella: clicchiamo col pulsante destro sul simbolino del DB che abbiamo creato. Scegliamo *Add New Table.*

3 AGGIUNGERE UN CAMPO - Creata la tabella, si aprirà la finestra mostrata in figura, nella quale possiamo inserire i campi della tabella.

```
MessageBox.Show("Ci sono "+DS.Tables[
                    "Articolo"].Rows.Count+" articoli");
```

Questo è ancora più utile quando si utilizzano *DataSet* tipizzati *(typed DataSet)* in cui vogliamo che i dati vadano in tabelle già esistenti. L'uso più interessante dell'accoppiata *DataAdapter/ DataSet* è però quello di fornire un mezzo semplice e potentissimo per l'inserimento e la modifica di righe in maniera disconnessa. Questa funzionalità è realizzata dal metodo *Update*, presente anche nel *MySqlDataAdapter.* Nell'esempio che segue andremo a recuperare i dati dalla tabella utente, aggiungeremo, toglieremo e modificheremo alcune righe, e poi chiederemo al *DataAdapter* di effettuare l'aggiornamento. Cominciamo con il creare la connessione e il *DataAdapter*:

```
MySqlConnection MyConn= new MySqlConnection(
   "server=localhost; database=marco1; user id=root;
                              password=snowball;");
MyConn.Open();
MySqlDataAdapter DA= new MySqlDataAdapter(
                      "select * from utente",MyConn);
```

per far sì che il DataAdapter sappia come fare insert, update e delete, dovremmo andare a creare i vari *MySqlCommand* membri del *DataAdapter* che dovranno realizzare le operazioni: l'*InsertCommand*, l'*UpdateCommand* e il *DeleteCommand*. (il *SelectCommand*, ovvero il command che estrae i dati, è creato in maniera automatica a partire dalla query)
Esiste però un metodo più veloce per far questo: utilizzare il *CommandBuilder.* Questo oggetto crea i suddetti *Command* per un dato *DataAdapter* a partire dalla select del *SelectCommand:*

```
MySqlCommandBuilder B=
                       new MySqlCommandBuilder(DA);
```

A questo punto siamo in grado di creare una *DataTable*, riempirla di dati e chiudere la connessione:

```
DataTable DT= new DataTable("Utente");
DA.Fill(DT);
MyConn.Close();
```

Ora abbiamo in locale, disconnessi, i dati della tabella *Utente* nella *DataTable DT.* Andiamo ad inserire due righe, modificarne una e cancellarne una quarta:

```
DT.Rows.Add(new object[]
          {"pippo","pippo@topolinia.top","oppip"});
DT.Rows.Add(new object[]
          {"paperino","pap@topolinia.top","rino"});
DT.Select("email='marco'")[0].Delete();
DT.Select("email='anna'")[0][1]="annina";
```

Per riportare queste modifiche sul DB non serve altro che riaprire la connessione ed aggiornare i dati:

```
MyConn.Open();
DA.Update(DT);
```

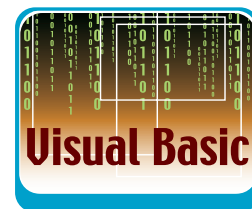
possiamo ora richiudere definitivamente la connessione:

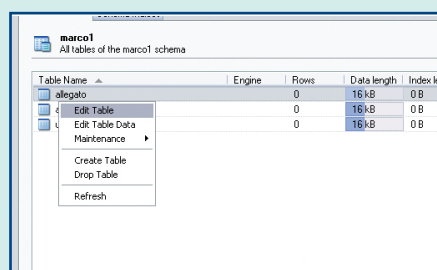
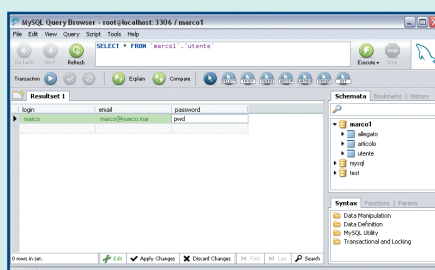
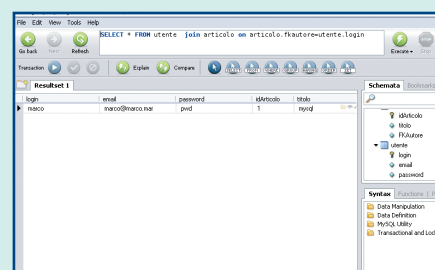
```
MyConn.Close();
```

CONCLUSIONI

Abbiamo visto quindi come è possibile utilizzare MySQL in maniera semplice, e soprattutto riutilizzando le tecniche e le conoscenze che abbiamo sviluppato per SQLServer.
Utilizzando il provider managed fornito da MySQL, o un altro simile, anche in .NET l'accesso ai dati in MySQL è divenuto prestazionale e semplicemente utilizzabile.

Marco Poponi

Visual Basic

NOTA

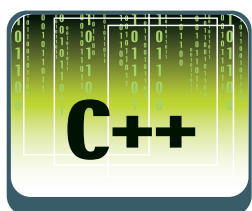
Si faccia attenzione nell'uso del *CommandBuilder*, in quanto per avere la struttura della tabella esso fa una query sul DB. Inoltre non ne consigliato l'uso nel caso ci sia bisogno di strategie di Update o Insert personalizzate.

4 MODIFICARE TABELLE ESISTENTI - Le tabelle create potranno essere modificate ciccando col pulsante destro sulla lista delle tabelle e scegliendo *Edit Table.*

5 INSERIRE DEI DATI - Utilizzando lo stesso menu, alla voce *Edit Table Data*, possiamo aprire *MySql Query Browser* per inserire dati. Quando abbiamo finito basterà cliccare su *ApplyChanges*.

6 ESEGUIRE QUERY - Un altro modo per inserire dati, e per estrarli è quello di scrivere direttamente la query SQL. Possiamo utilizzare il MySql Query Browser.

Creiamo da zero un ambiente 3D in stile Quake

Ambienti 3D con IrrLicht

parte 2

In questa puntata creeremo un programma che permetta di caricare, visualizzare ed esplorare uno spazio 3D. Utilizzeremo il C++, IrrLicht e una mappa compatibile col noto videogioco Quake 3 Arena

Utilizza questo spazio per le tue annotazioni

La potenza di un motore grafico può essere definita come il rapporto tra la quantità di cose che riesce a fare e lo sforzo richiesto per farle. IrrLicht, da questo punto di vista, è un motore molto potente. Riesce infatti a rendere semplici cose effettivamente molto complesse da codificare "a mano". Il tutto senza perdere di vista "il mondo reale", popolato da formati di file che sono standard di fatto e algoritmi utilizzati un po' dappertutto nel medesimo modo. In questo articolo vedremo come sia facile realizzare lo scheletro di un *FPS (= First Person Shooter)*. Utilizzeremo una mappa creata per Quake 3 un pugno di righe di codice, e IrrLicht una libreria opensource.

ESTENDIAMO IL FILESYSTEM

Nello scorso numero di ioProgrammo abbiamo creato la struttura base di un gioco 3D. Per chi si fosse perso questa puntata si faccia riferimento al box *"Lo scheletro del programma"* Quello che dobbiamo fare è caricare la mappa che andremo a esplorare e aggiungere le relative istruzioni di caricamento nella porzione di inizializzazione del motore. Per farlo dobbiamo prima compiere una semplice operazione. Semplice da fare, ma che ci fa ben comprendere quale sia la potenza di IrrLicht con questo genere di cose. Utilizzando questo motore possiamo "estendere" il file system del PC, aggiungendo dei file in forma compressa. In altre parole quando abbiamo a che fare con archivi di materiale multimediale come file grafici per texture, file sonori o file di configurazione, possiamo accedervi semplicemente "attaccando" l'archivio al file system. Questa operazione viene compiuta con una sola linea di codice. Nel nostro caso aggiungeremo il file *"map-20kdm2.pk3"* contenente la mappa che ci interessa. Il formato *.pk3* non è altro che il formato *.zip* dei comuni archivi compressi. È possibile trovare, per qualsiasi "prova sul campo", questo file sotto la cartella "media" della cartella principale di IrrLicht. Aggiungiamo il contenuto di questo archivio ai file "visibili" dal nostro programma con la seguente istruzione:

```
device->getFileSystem()->addZipFileArchive(
                                        "map-20kdm2.pk3");
```

Come si può vedere tutta la parte di decompressione e i relativi algoritmi sono "trasparenti" al programmatore, che si può concentrare semplicemente sul contenuto dell'archivio, senza sapere come questo viene reso disponibile.

CARICHIAMO LA MAPPA

Una volta attaccato l'archivio compresso al file system, possiamo leggerne il contenuto esattamente come se si trattasse di file presenti sul nostro hard disk in forma "espansa". All'interno dell'archivio in questione è presente la mappa vera e propria. Il no-

Fig. 1: **Un dettaglio della mappa caricata**

me della mappa è *"20kdm2.bsp"*. L'estensione *.bsp* sta per *"Binary Space Partitioning"* e indica un formato pre-compilato, in cui tutte le informazioni geometriche sono sistemate in maniera tale da rendere semplice l'esclusione dalla fase di disegno (rendering) delle parti di mappa che sicuramente non sono visibili. Evitare di effettuare calcoli per zone non visibili consente di mantenere alto il frame-rate, cioè il numero di fotogrammi per secondo disegnati a schermo. Implementare "manualmente" un algoritmo *bsp* e il relativo formato dati sarebbe un compito tutt'altro che banale. In IrrLicht tutto ciò è fornito "gratuitamente" allo sviluppatore: niente male eh? Il caricamento vero e proprio della mappa avviene con le seguenti righe di codice:

```
IAnimatedMesh* mesh = smgr-> getMesh("20kdm2.bsp");
ISceneNode* node = 0;
if (mesh)
  node = smgr->addOctTreeSceneNode(mesh->
                                  getMesh(0));
```

Viene creata una struttura geometrica (mesh) tramite la funzione *getMesh()* dello scene manager. Questa funzione accetta in ingresso una stringa contenente il nome del file che contiene la mesh. Questo file è a sua volta contenuto nell'archivio aggiunto in precedenza, ma possiamo tranquillamente dimenticarci di questa cosa. Lo scene manager viene utilizzato per "agganciare" la mesh appena creata alla scena corrente, al fine di poterla disegnare a video quando sarà invocata la *drawAll()*, nel ciclo principale del programma. Per questo compito viene utilizzata la funzione *addOctTreeSceneNode()*. Questa

LO SCHELETRO DEL PROGRAMMA

Costruire la struttura base di un programma-tipo che utilizzi IrrLicht è semplicissimo: basta seguire questi passi:

1 INCLUSIONI - Inseriamo nel sorgente del nostro programma le seguenti direttive:

```
#include <irrlicht.h>
using namespace irr;
using namespace core;
using namespace scene;
using namespace video;
using namespace io;
using namespace gui;
#pragma comment(lib, "Irrlicht.lib")
```

Queste servono per dire al compilatore quale è il file header da considerare per utilizzare IrrLicht ("*irrlicht.h*"); quali sono i namespace usati (*irr, core, scene, video, io* e *gui*) e qual'è la libreria da impiegare per il link ("*irrlicht.lib*"). A runtime sarà utilizzata la relativa libreria dinamica "*irrlicht.dll*".

2 INIZIALIZZAZIONE - Le seguenti linee di codice istanziano il device, il driver video e lo scene manager, che sono oggetti essenziali per il funzionamento di IrrLicht.

```
IrrlichtDevice *device =createDevice(
        EDT_DIRECTX8, dimension2d<s32>(640, 480),
                        16,false, false, false, 0);
IVideoDriver* driver = device->getVideoDriver();
ISceneManager* smgr = device->getSceneManager();
```

Viene creato un device di visualizzazione in 640x480 pixel, utilizzando le librerie multimediali DirectX 8 e una profondità di colore di 16 bit. I puntatori a *IVideoDriver* e *ISceneManager* servono per la gestione della scena, che ovviamente varia da programma a programma. Il relativo codice va comunque aggiunto qui.

3 CICLO PRINCIPALE - Quasi ogni programma che preveda l'interazione dell'utente con un ambiente real-time presenta un ciclo principale. Questo si occupa di gestire gli input che riceve dall'esterno e disegnare di conseguenza la scena 3D. IrrLicht non fa eccezione e questo è il suo ciclo principale:

```
while(device->run()) {
        driver->beginScene(true, true,
SColor(0,200,200,200));
        smgr->drawAll();
        driver->endScene();
}
```

Il driver dà inizio alla scena e la conclude, lo *Scene Manager (smgr)* si occupa di stabilire cosa deve essere manipolato e/o disegnato. In questo caso si disegna tutto (cioè nulla!) tramite la funzione *drawAll()*.

4 CLEAN-UP - Il ciclo precedente viene interrotto quando il device non è più in esecuzione, ovvero quando la condizione *run()* diventa falsa. Questo avviene quando ad esempio si preme *ALT+F4* per terminare il programma. Prima di uscire è buona norma rilasciare tutte le risorse utilizzate. Questa fase è gestita in IrrLicht tramite la seguente chiamata:

```
device->drop();
```

Tutte le fasi di pulizia e deallocazione della memoria sono automatiche e gestite in maniera trasparente dal motore. Ovviamente in questa fase saranno inserite anche tutte le operazioni di rilascio di oggetti creati al di fuori di IrrLicht.

EDITOR DI MAPPE

Le mappe utilizzabili con *IrrLicht* sono le stesse che funzionano con Quake 3 Arena. In rete sono disponibili numerosi tool, liberamente scaricabili, che consentono di costruire da zero una mappa di questo tipo. Uno di questi tool, probabilmente il più famoso, è GTK Radiant. Lo potete trovare all'indirizzo http://www.qeradiant.com nelle sue versioni per Windows e Linux. Allo stesso indirizzo sono presenti anche dei tutorial e una buona dose di informazioni utili per il suo apprendimento.

LA LICENZA

Molto spesso gli strumenti per la realizzazione di software multimediale, vengono venduti a peso d'oro agli sviluppatori. *IrrLicht* è completamente gratuito, oltre ad essere open source. Non solo: è liberamente modificabile in ogni sua parte senza il vincolo di rendere disponibile il codice delle modifiche. Come se non bastasse è liberamente utilizzabile in progetti di tipo commerciale, senza obbligo di corrispondere alcuna royalty agli sviluppatori. Questi richiedono solo di essere citati tra i "credits" del progetto sviluppato.

OCTTREE E BSP

Un *OctTree* è una struttura dati che definisce una forma tridimensionale ottimizzandone il contenuto. Ogni porzione di spazio è divisa in cubi. A sua volta ogni cubo può essere diviso in 8 cubi più piccoli uguali tra loro. Allo stesso modo un BSP divide lo spazio in entità binarie. Laddove è necessaria una maggiore precisione l'entità è divisa in due, in quattro, in otto e così via. *OctTree* e *BSP* sono alla base dei motori che gestiscono grafica 3D in tempo reale.

funzione consente di utilizzare la struttura *bsp* caricata con la mesh, e di usufruire dei relativi vantaggi in termini di velocità. È possibile utilizzare, ottenendo gli stessi risultati per quanto riguarda la visualizzazione, anche la funzione *addAnimatedMeshSceneNode()*. Questa funzione tuttavia gestisce tutta la struttura geometrica caricata, senza eliminare le parti non visibili e per questo risulta più lenta.
Il nodo *(ISceneNode* node)* aggiunto alla scena principale contiene la mesh ottenuta tramite la funzione *getMesh()*. Il parametro numerico 0, passato come argomento, indica che andrà preso il primo frame di animazione della mesh. Tuttavia, in questo caso, la mesh della mappa non ha animazioni, per cui vi è un solo frame e non possiamo che prendere quello. Nelle prossime puntate spiegheremo come gestire le mesh con più di un frame di animazione.

CREIAMO LA VIDEOCAMERA

Per fare si che venga visualizzato qualcosa a schermo, occorre aggiungere alla scena corrente un nodo che contenga una videocamera. Le videocamere in IrrLicht possono essere di differenti tipi: da quelle semplici che forniscono solamente la visualizzazione, a quelle più complesse, che gestiscono anche gli input da tastiera. Utilizzeremo in questo caso una videocamera di quest'ultimo tipo. Siccome quello che abbiamo in mente è un FPS, possiamo aggiungere con la seguente riga di codice

```
ICameraSceneNode* camera  = smgr->
                              addCameraSceneNodeFPS();
```

una videocamera che ci consente di muovere la visuale in maniera simile a quella di molti videogiochi in prima persona. Potremo controllare la direzione dello sguardo muovendo il mouse, mentre ci sposteremo in avanti e indietro con le frecce *SU* e *GIU* della tastiera. Sempre con le frecce (*DESTRA* e *SINISTRA*) potremo effettuare lo "scorrimento" laterale (strafe). Già che ci siamo rendiamo migliore l'aspetto del nostro programma eliminando il cursore del mouse, che altrimenti resterebbe fisso al centro della finestra di visualizzazione. L'istruzione per fare ciò è

Fig. 2: Le videocamere ci consentono di modificare la prospettiva della visualizzazione

```
device->getCursorControl()->setVisible(false);
```

Una possibile alternativa potrebbe essere quella di visualizzare un cursore differente dalla classica freccia bianca di Windows. In molti giochi, ad esempio, è presente una piccola croce che simula il mirino di un'arma. Nel nostro caso abbiamo preferito la semplicità, anche perché non implementeremo, per ora, alcuna arma con cui sparare! In ogni caso il primo passo per la gestione del cursore è quello di ottenere un puntatore al relativo controllore, tramite la funzione *getCursorControl()*.

GESTIRE LE COLLISIONI

Se provassimo il codice così com'è quello che otterremmo sarebbe la possibilità di muoverci all'interno della mappa "a volo d'angelo". In altre parole sarebbe possibile spostarsi in qualsiasi punto della mappa, a qualsiasi altezza, passando anche attraverso muri. Per evitare questo comportamento un po' "naive" e dare il senso di una reale "camminata", dobbiamo in qualche modo gestire le collisioni. Il programma deve, in breve, reagire all'impatto della videocamera con i muri, bloccando lo spostamento e simulando così la presenza di un ostacolo fisico. Fortunatamente, come avrete intuito, la gestione di collisioni in IrrLicht è un argomento abbastenza "leggero". Sebbene infatti questo compito richieda l'implementazione di algoritmi complessi, utilizzando il motore grafico in questione, troviamo già tutto realizzato e pronto all'uso.
Quello che bisogna fare innanzitutto è creare un oggetto di tipo *ITriangleSelector.* Questi oggetti sono utilizzati in IrrLicht per diversi scopi: uno di questi è proprio la gestione delle collisioni. Aggiungiamo quindi al nostro sorgente le seguenti righe di codice:

```
ITriangleSelector* selector = 0;
selector = smgr->createOctTreeTriangleSelector(
mesh->getMesh(0), node, 128);
```

Abbiamo creato un selettore di tipo *OctTree.* L'unica cosa che ci interessa sapere in questo caso è che questo tipo di selettore è ottimizzato per lavorare congiuntamente con i file *bsp.* Proprio il tipo di file che abbiamo aggiunto alla scena principale, sempre tramite una funzione di tipo *OctTree.* Come linea generale utilizzando una mesh *OctTree* utilizzeremo un selettore dello stesso tipo. Molto semplice.

SCENE ANIMATORS

Siamo a un passo dal completare l'opera. L'ultima cosa da fare è aggiungere alla videocamera FPS-like, creata in precedenza, quello che viene chiamato un "animatore". I villaggi vacanza in questo caso non c'entrano: un animatore è semplicemente un ogget-

to di IrrLicht che si occupa di conferire al nodo della scena cui è assegnato un comportamento. L'animatore in questione utilizzerà il selettore creato in precedenza per gestire le collisioni della videocamera con la mesh della mappa:

```
ISceneNodeAnimator* anim = smgr->
    createCollisionResponseAnimator(selector, camera,
    vector3df(30,50,30), vector3df(0,-100,0), 100.0f,
    vector3df(0,50,0));
camera->addAnimator(anim);
anim->drop();
```

La funzione di creazione *createCollisionResponseAnimator()* accetta in ingresso ben sei parametri, vediamo il loro significato:

- **parametro 1**: il selettore per la gestione delle collisioni
- **parametro 2**: l'oggetto che verrà gestito da questo animatore. Nel nostro caso la videocamera.
- **parametro 3**: la dimensione dell'oggetto passata tramite un *vector3df*
- **parametro 4**: la direzione dell'accelerazione di gravità
- **parametro 5**: il valore dell'accelerazione di gravità
- **parametro 6**: la traslazione del punto centrale dell'oggetto rispetto al nodo cui è assegnato

Una volta creato, l'animatore viene aggiunto alla videocamera tramite la funzione *addAnimator()*. Fatto questo è possibile (anzi consigliato) eliminare il riferimento all'animatore appena creato tramite la funzione *drop()*. L'animatore continuerà ad esistere in quanto una sua copia è stata assegnata a un oggetto attivo nella scena.

IL CODICE COMPLETO

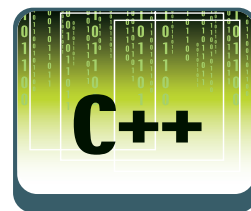
Quanto sinora descritto completa in linea di massima l'applicazione che ci siamo prefissati di realizzare. Il codice completo del programma è riportato di seguito:

```
int main() {
  //Inizializzazione
  IrrlichtDevice *device = createDevice(EDT_DIRECTX8,
      dimension2d<s32>(640, 480), 16,
      false, false, false, 0);
  IVideoDriver* driver = device->getVideoDriver();
  ISceneManager* smgr = device->getSceneManager();
  device->setWindowCaption(L"Ambienti 3D con IrrLicht (
      parte 2)");
  //Caricamento della mappa
  device->getFileSystem()->addZipFileArchive(
      "map-20kdm2.pk3");
  IAnimatedMesh* mesh = smgr->getMesh(
      "20kdm2.bsp");
  ISceneNode* node = 0;
  if (mesh)
    node = smgr->addOctTreeSceneNode(mesh->
        getMesh(0));
    //Creazione della videocamera
    ICameraSceneNode* camera = smgr->
        addCameraSceneNodeFPS();
    camera->setPosition(vector3df(1100,700,1500));
    device->getCursorControl()->setVisible(false);
    //Gestione delle collisioni
    ITriangleSelector* selector = 0;
    selector = smgr->createOctTreeTriangleSelector(
    mesh->getMesh(0), node, 128);
    ISceneNodeAnimator* anim = smgr->
     createCollisionResponseAnimator(selector, camera,
       vector3df(30,50,30), core::vector3df(0,-100,0),
       100.0f, core::vector3df(0,50,0));
    camera->addAnimator(anim);
    anim->drop();
    //Ciclo fondamentale del programma: disegno
        della scena
    while(device->run())  {
      driver->beginScene(true, true, SColor(
          0,200,200,200));
      smgr->drawAll();
      driver->endScene(); }
    //Rilascio delle risorse
    device->drop();
    return 0;
}
```

Come si può vedere poco più di 20 istruzioni sono necessarie per creare un programma la cui codifica da zero richiederebbe diverse settimane di lavoro. Tutte le istruzioni che abbiamo aggiunto alla struttura base del codice sono state inserite nella parte di "inizializzazione", mentre il ciclo principale è rimasto inalterato. Questa è una delle possibili soluzioni, tuttavia è anche possibile manipolare la scena all'interno del loop principale, qualora ci sia qualche modifica da effettuare "al volo".

CONCLUSIONI

In questa puntata abbiamo visto come sia possibile creare un mini-fps con pochissimo sforzo utilizzando IrrLicht. Abbiamo imparato a caricare una mappa in formato compresso e ottimizzato e abbiamo visto come creare una videocamera controllata da tastiera. Sono state inoltre gestite le collisioni con l'ambiente circostante, ricreando, a grandi linee, il comportamento di un videogioco in prima persona. Nei prossimi numeri continueremo a occuparci di IrrLicht, non mancate!

Alfredo Marroccelli

IRRLICHT IN RETE

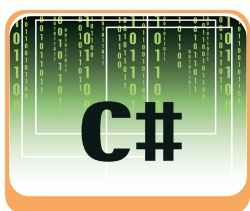
È possibile trovare ogni informazione disponibile su IrrLicht all'indirizzo ufficiale http://irrlicht.sourceforge.net. Attorno a questo progetto ambizioso è sorta una discreta comunità di utilizzatori e nascono quotidianamente progetti interessanti, da giochi ad applicazioni più "serie". Non mancate di farci un salto!

PROGETTI IN IRRLICHT

Questo potente motore grafico non ha mancato di suscitare l'attenzione di molti appassionati di programmazione 3D in giro per il mondo, sin dalle sue prime release. Sono sorti così diversi progetti, anche molto corposi, reperibili in rete. Questi progetti costituiscono, oltre che un test delle potenzialità del motore, anche un ottimo banco di prova per apprendere i "trucchi del mestiere", analizzando il codice sorgente disponibile. Potete trovare informazioni aggiuntive nella sezione "Project Announcements" del forum di IrrLicht, raggiungibile da http://irrlicht.sourceforge.net/phpBB2/index.php.

.NET e il codice umanaged pronti per la steganografia

Non aprite quell'immagine!!!

Vedremo cosa è la steganografia, ne analizzeremo gli aspetti fondamentali, realizzeremo un progetto che ci permetterà di nascondere un messaggio all'interno di un'immagine

Utilizza questo spazio per le tue annotazioni

Tramite la steganografia le informazioni possono essere nascoste all'interno di insospettabili "contenitori" (detti *cover*), quindi è un argomento che ricopre uno spazio importantissimo nell'ambito della sicurezza informatica e non solo.

TECNICHE STEGANOGRAFICHE

Sostanzialmente ci sono tre metodi per nascondere un messaggio all'interno di un cover:

- **Iniezione:** tale metodo consiste nell'iniettare i bit relativi al messaggio da nascondere all'interno del cover.
  La cosa negativa, utilizzando questo metodo, è che generalmente fa aumentare notevolmente la grandezza del cover (in termini di byte).

- **Sostituzione:** essa è utilizzata per rimpiazzare i bit "inutili" del cover con i bit che costituiscono il messaggio. Il problema che potrebbe sorgere usando questo metodo, è che un suo abuso potrebbe far degradare notevolmente la qualità del cover insospettendo una terza parte che viene in possesso dello "stego file" (cioè il *cover* con all'interno il messaggio segreto) e questo "terzo incomodo" potrebbe essere benissimo uno *steganalista*.

- **Generazione di nuovi file:** in questo caso il cover inizialmente non esiste, ma viene generato con il solo scopo di nascondere il messaggio, cioè viene generato il cover per "wrappare" il messaggio da nascondere.

IMPLEMENTAZIONE

I tre metodi utilizzati in campo steganografico, cioè iniezione, sostituzione e generazione di nuovi file, vengono implementati mediante sei tecniche:

1. Tecniche di sostituzione
2. Tecniche del dominio della trasformata
3. Tecniche di spread spectrum
4. Tecniche che usano i metodi statistici
5. Tecniche di distorsione
6. Tecniche della generazione dei cover

Di queste, per ragioni di spazio, entreremo nel dettaglio solo della tecnica di sostituzione in quanto è quella che utilizzeremo nel nostro progetto.

UN PO' DI TEORIA

Tramite la tecnica della sostituzione, il messaggio viene nascosto rimpiazzando i bit ridondanti (vedi box) del cover con i bit del messaggio da nascondere. Il metodo più utilizzato è il cosiddetto metodo *LSB* (*Least Significant Bit*, cioè *Bit Meno Significativo*). Vediamo come funziona tale metodo un po' più in dettaglio delle atre tecniche in quanto è il metodo che useremo nel nostro progetto. Tanto per cominciare chiariamo il concetto di *LSB*. È il bit meno significativo di un byte. Gli 8 bit che costituiscono un byte vanno da sinistra verso destra in ordine di importanza. Ad esempio nel byte rappresentato da questa sequenza di bit:

*1001001**0***
l'ultimo zero (quello indicato in grassetto) è l'LSB.

Il metodo *LSB* consiste nel sostituire l'*LSB* di alcuni byte del cover in modo da rimpiazzarli con i bit che costituiscono il messaggio da nascondere.

Ad esempio, supponiamo che la seguente sequenza di byte costituisca parte del nostro cover:

...

*1000100**1***	*1100001**1***
*0101000**1***	*0011100**1***
*0111110**0***	*1111001**1***
*1101101**0***	*0101111**0***

...

e supponiamo di voler nascondere il carattere a in questa sequenza di byte. Il carattere a in codice ASCII è rappresentato dal numero decimale 97 che in binario corrisponde a:

01100001

Utilizzando il metodo LSB (useremo l'ultimo bit di ogni byte del cover) inseriremo ora il carattere a nella sequenza di byte del nostro cover:

- 1000100**1:** L'*1* verrà sostituito dallo *0*, cioè il primo bit del nostro messaggio
- 1100001**1:** L'*1* verrà lasciato tale in quanto corrisponde già al secondo bit del nostro messaggio
- 0101000**1:** L'*1* verrà lasciato tale in quanto corrisponde già al terzo bit del nostro messaggio
- 0011100**1:** L'*1* verrà sostituito dallo *0*, cioè il quarto bit del nostro messaggio
- 0111110**0:** Lo *0* verrà lasciato tale in quanto corrisponde già al quinto bit del nostro messaggio
- 1111001**1:** L'*1* verrà sostituito dallo *0*, cioè il sesto bit del nostro messaggio
- 1101101**0:** Lo *0* verrà lasciato tale in quanto corrisponde già al settimo bit del nostro messaggio
- 0101111**0:** Lo *0* verrà sostituito dall'*1*, cioè l'ottavo e ultimo bit del nostro messaggio.

Come si può vedere abbiamo inserito il nostro messaggio (in questo esempio costituito solo dalla lettera a) all'interno dei byte del nostro cover modificando solo 4 degli 8 byte. Questa è una cosa da tenere presente perché molto interessante. Cioè per inserire il nostro messaggio all'interno del cover non è detto che dovremo modificare un byte del cover per ogni bit del nostro messaggio, infatti si può verificare che l'*LSB* del cover coincida col bit in questione del messaggio da nascondere, come infatti è successo nell'esempio che abbiamo fatto, degradando le caratteristiche originali del cover in minor misura.

IL NOSTRO PROGETTO

Come già accennato, per il nostro progetto utilizzeremo la tecnica della sostituzione e in particolare il metodo *LSB*. Come cover useremo delle immagini nel formato *24 bpp*. Le **Figure 1** e **2** mostrano l'architettura di una generica applicazione di steganografia. Come si può vedere in **Figura 1**, per quanto riguarda la codifica, in input abbiamo:

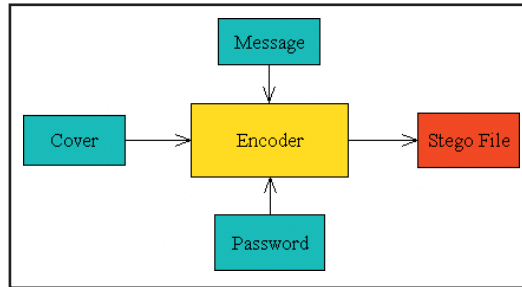



Fig. 1: ***Input e Output dell'Encoder***

- Il cover in cui verrà inserito il messaggio
- Il messaggio vero e proprio
- La password utilizzata per criptare il messaggio prima di inserirlo nel cover, in modo da avere un'ulteriore sicurezza

L'output è rappresentato dallo *"stego file"*, cioè il cover originale con i necessari bit rimpiazzati in modo da contenere il nostro messaggio. Molti di voi rimarranno sorpresi del notare che lo *stego file* sarà "uguale", almeno dal punto di vista ottico, al cover. In realtà, l'immagine contenente il messaggio sarà "quasi" uguale all'immagine originale. Diciamo "quasi" perché se si fa un confronto a livello di bit ovviamente non sarà così, però dal punto di vista ottico tale differenza non sarà notata.

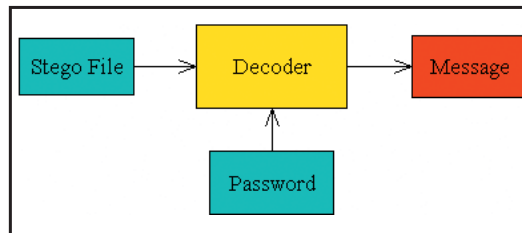



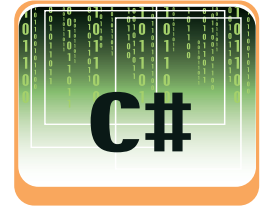
Fig. 2: ***Input e Output del Decoder***

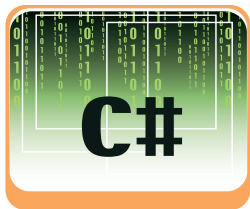
Per quanto riguarda la decodifica invece (**Figura 2**), in input abbiamo:

- Lo stego file da cui verrà estratto il messaggio
- La password che è stata utilizzata per la codifica

ottenendo in output il messaggio vero e proprio.

DALLE PAROLE AI FATTI

Per motivi di spazio non verranno elencati tutti i listati del progetto. Sul CD allegato alla rivista, tuttavia, è presente il codice completo ampiamente commentato, con in più una semplice ma efficace interfaccia grafica che potrete cambiare a vostro piacimento in quanto, come vedremo, sarà indipendente

UN PO' DI STORIA

I padri della steganografia possono essere considerati gli egiziani che con i loro geroglifici hanno creato la prima forma di "scrittura nascosta". Infatti, agli occhi di chi non sapeva come interpretarli, quei simboli erano dei semplici "disegni". Una forma di steganografia un po' più "recente" è quella dei micropunti fotografici: si tratta di fotografie della dimensione di un punto che, una volta sviluppate e ingrandite, rivelano il messaggio nascosto. Tale metodo venne massicciamente utilizzato dai nazisti durante la Seconda Guerra Mondiale.

NOTA

DIFFERENZA TRA STEGANOGRAFIA E CRITTOGRAFIA

La principale differenza tra la steganografia e la crittografia è che nella prima il file trasmesso sembra un file qualunque agli occhi di un ignaro osservatore mentre nella crittografia l'esistenza di un messaggio nascosto è evidente. La forza della steganografia quindi sta nel fatto che l'esistenza del messaggio nascosto da trasmettere è occultata.

GLOSSARIO

BIT RIDONDANTI

Per bit ridondanti di un cover si intende quei bit che, anche se modificati, non alterano drasticamente le caratteristiche del cover. Per fare un esempio col mondo reale, se il nostro conto in banca è di 1.000.001 euro e cambiamo l'ultima cifra in 0 ottenendo 1.000.000 di euro, di sicuro non abbiamo cambiato "drasticamente" le caratteristiche del nostro conto...cioè la carta d'oro non ce la tolgono di sicuro!

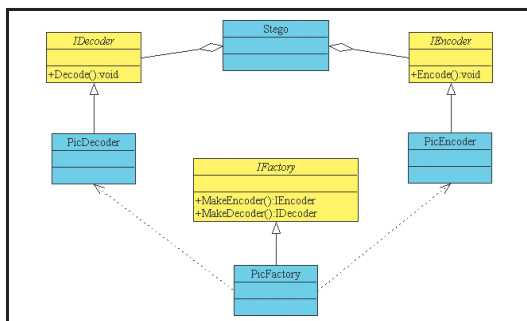



Fig. 3: ***Class Diagram dell'applicazione***

dal progetto. Il *Class Diagram* (un po' semplificato) relativo al progetto è mostrato in **Figura 3**. Come si può ben vedere abbiamo utilizzato lo *Strategy Pattern* e l'*Abstract Factory Pattern*. Per chi non conosce tali pattern nessun problema, tra qualche riga capirete di cosa si tratta. La necessità di utilizzare le interfacce *IEncoder* e *IDecoder* sta nel fatto che se un domani decidessimo di ampliare il nostro progetto, non dovremo fare dei grossi cambiamenti al codice. Ad esempio se decideremo di implementare un algoritmo steganografico che utilizzi dei file audio come cover invece di immagini, basterà creare due nuove classi, una che implementi l'interfaccia *IEncoder* e l'altra che implementi *IDecoder*.
Ad esempio, chiamando tali classi *WavEncoder* e *WavDecoder* avremo:

```
public class WavEncoder : IEncoder {
...
  public void Encode(string cover, Stream message,
                                     string password) {...}
...
}
public class WavDecoder : IDecoder {
...
  public void Decode(string stegoFile, string password) {...}
...
}
```

La classe *Stego* saprà di avere a che fare con riferimenti del tipo *IEncoder* e *IDecoder* prescindendo dalla particolare implementazione *(Strategy Pattern)*. Grazie all'uso dell'interfaccia *IFactory*, invece, creeremo gli oggetti "giusti", di tipo *IEncoder* e *IDecoder*, da dare "in pasto" alla classe *Stego (Abstract Factory Pattern)*. Ad esempio nel caso di steganografia che utilizza le immagini come cover avremo un qualcosa del genere:

```
IFactory factory = new PicFactory();
Stego stego;
// per la codifica
IEncoder encoder = factory.MakeEncoder();
stego = new Stego(encoder);
...
// per la decodifica
IDecoder decoder = factory.MakeDecoder();
stego = new Stego(decoder);
```

Ovviamente nel caso in cui si voglia implementare l'algoritmo che utilizza come cover i file audio dovremo creare, oltre alle classi *WavEncoder* e *WavDecoder*, anche una classe *WavFactory* che implementa *IFactory* e che si occuperà di creare oggetti di tipo *WavEncoder* e *WavDecoder*.

INSERIMENTO DEL MESSAGGIO SEGRETO ALL'INTERNO DELL'IMMAGINE

Per occultare il messaggio all'interno dell'immagine utilizzeremo la classe *PicEncoder*. Per l'implementazione di questa classe e della classe *PicDecoder* abbiamo utilizzato codice unmanaged, in quanto, per ragioni legate alle performance, abbiamo fatto uso dei puntatori per accedere ai byte rappresentanti i pixel dell'immagine. Per la classe *PicEncoder*, come molti avranno immaginato, il metodo che si occupa dell'inserimento del messaggio nell'immagine è *Encode*. Vediamone il codice:

```
public void Encode(string cov, Stream mes, string pw) {
  Bitmap bitmap = new Bitmap(cov);
  cover = new PicCover(bitmap);
  maxWidth = this.cover.Pic.Width * 3;
  message = new Text(mes);
  password = pw;
  if(!OkLength())
    throw new Exception("Il testo è troppo lungo per
                                    l'immagine selezionata!);
  Stream encryptedMessage = Crypt.EncryptDecrypt(
                                 message.Testo, password);
  int testoLength = (int) encryptedMessage.Length;
  byte[] testoLengthToByte = BitConverter.GetBytes(
                                             testoLength);
  Stream encryptedLength = Crypt.EncryptDecrypt(new
        MemoryStream(testoLengthToByte), password);
  testoLengthToByte = Read(encryptedLength);
  LockBitmap();
  Replace(testoLengthToByte);
  encryptedLength.Close();
  byte[] testoToByte = Read(encryptedMessage);
  Replace(testoToByte);
  encryptedMessage.Close();
  UnlockBitmap();
}
```

Analizziamo tale codice. Come si può ben vedere il metodo riceve in ingresso:

- una stringa rappresentante il percorso in cui si

trova il cover, cioè la nostra immagine
- uno stream rappresentante il messaggio da nascondere
- una stringa rappresentante la password da utilizzare per criptare il messaggio prima di nasconderlo

Le prime cinque righe di codice servono a valorizzare i campi privati della classe *PicEncoder* con i valori ricevuti in ingresso. Dopodiché viene controllato che il messaggio non sia troppo lungo per essere contenuto nell'immagine. Se tutto è andato a buon fine allora il messaggio viene criptato utilizzando la classe *Crypt* che vedremo tra poco. A questo punto viene calcolata la lunghezza del testo da nascondere e convertita in un array di byte. Tale lunghezza, per una maggiore sicurezza, viene poi criptata. Qui arriva una parte un po' più delicata. Dato che abbiamo utilizzato codice unmanaged per accedere ai pixel dell'immagine, vi è la necessità di "bloccarla" in memoria. Questo perché altrimenti potrebbe venire spostata dal garbage collector causando possibili riferimenti errati da parte del puntatore. Cioè il nostro puntatore "penserebbe" di puntare ancora all'immagine quando invece non è così. Allo scopo di bloccare l'immagine in memoria abbiamo utilizzato il metodo *LockBitmap* così implementato:

```
private void LockBitmap() {
  Rectangle bounds = new Rectangle(0, 0,
                          cover.Pic.Width, cover.Pic.Height);
  bitmapData = cover.Pic.LockBits(bounds,
                                 ImageLockMode.ReadWrite,
                              PixelFormat.Format24bppRgb);
  pointer = (Byte*)(bitmapData.Scan0.ToPointer());}
```

LockBitmap blocca tutti i pixel dell'immagine tramite il metodo *LockBits* facente parte della classe *System.Drawing.Bitmap*. Tale metodo riceve in ingresso:

- una struttura di tipo Rectangle che indica la porzione di immagine da bloccare (nel nostro caso tutta)
- l'enumerazione *ImageLockMode* indica il livello di accesso ai pixel dell'immagine (nel nostro caso lettura e scrittura)
- l'enumerazione *PixelFormat* indica il formato dell'immagine (nel nostro caso 24 bit per pixel)

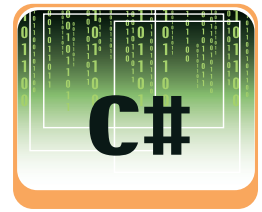
LockBits restituisce un oggetto di tipo *System.Drawing.Imaging.BitmapData*. Tale classe specifica gli attributi di un'immagine. Tramite l'ultima istruzione poi otteniamo un puntatore al primo byte del primo pixel dell'immagine. A questo punto vengono inseriti nell'immagine sia la lunghezza del testo sia il testo vero e proprio e ciò viene fatto tramite il metodo *Replace*. Vediamone l'implementazione:

```
private void Replace(byte[] message) {
  int offset = maxWidth % 4;
    for(int i = 0; i < message.Length; i++){
      byte b = message[i];
        for(int j = 0; j < 8; j++){
            Bit.Replace(ref *pointer, 7, Bit.Extract(b, j));
            pointer++;
            contatore++;
            if(offset != 0 && contatore == maxWidth){
              pointer += 4 - offset;
             contatore = 0;}
    } }
}
```

Tale metodo si occupa di inserire l'array di byte, passato come parametro, all'interno dei pixel dell'immagine (usando i metodi della class Bit che vedremo più avanti). Molti di voi avranno notato l'escamotage utilizzato per far "saltare" il puntatore di (*4 – offset*) byte nel caso in cui questo si trovi alla fine della linea che costituisce una riga dell'immagine. Questo è necessario in quanto il numero di byte che costituisce una linea di un'immagine deve essere sempre un multiplo di *4*, se così non fosse allora tale linea verrà "paddata" con *(4 – offset)* byte in modo da soddisfare tale requisito. Questa è una cosa molto importante da tenere sempre a mente quando si scrivono applicazioni che operano su immagini in quanto eviterà molti grattacapi. Per concludere l'immagine viene "sbloccata" dalla memoria tramite il metodo *UnlockBitmap*. Ecco il codice:

```
private void UnlockBitmap() {
  cover.Pic.UnlockBits(bitmapData);
  bitmapData = null;
  pointer = null;}
```

Lo sblocco viene fatto semplicemente chiamando il metodo *UnlockBits* della classe *System.Drawing.Bitmap*, passandogli in ingresso l'oggetto *bitmapData* visto precedentemente. Quella di seguito invece è l'implementazione della classe *Crypt:*

```
public class Crypt {
  public static Stream EncryptDecrypt(Stream stream,
                                      string password) {
    int passwordLength = password.Length;
    if(passwordLength == 0)
      password = "ioProgrammo";
    byte[] salt = {31, 1, 76};
    PasswordDeriveBytes pdb = new
                    PasswordDeriveBytes(password, salt);
    byte[] key = pdb.GetBytes(128);
    int streamLen = (int) stream.Length;
    int keyLength = key.Length;
    int oldByte;
    int counter = 0;
    MemoryStream copia=new MemoryStream(streamLen);
```

CODICE UNMANAGED

Il codice unmanaged è detto così perché gira al di fuori del CLR (Common Language Runtime) e quindi non è gestito (managed) da quest'ultimo. Quando facciamo uso dei puntatori in C#, .NET richiede che il codice da noi usato venga "marcato" da una parola chiave speciale. Tale parola chiave è *unsafe*. Così facendo il compilatore capisce che siamo coscienti delle nostre azioni. Tale "responsabilità" ce la dobbiamo inoltre prendere anche in fase di compilazione usando, il flag */unsafe* da riga di comando, o specificandolo nelle proprietà del progetto usando Visual Studio .NET.

IMMAGINI 24BPP

Le immagini 24bpp (24 bit per pixel) sono immagini che utilizzano 8 bit (cioè 1 byte) per ogni componente RGB (Red Green Blue) che esprime il colore del pixel, producendo 16.777.216 possibili colori.

```
    while((oldByte = stream.ReadByte()) != -1) {
      int index = counter % keyLength;
                    byte b = (byte) oldByte;
                    byte newByte = (byte)(key[index] ^ b);
                    copia.WriteByte(newByte);
                    counter++;}
    copia.Seek(0, SeekOrigin.Begin);
    return copia; }
}
```

OPERATORI ORIENTATI AI BIT

Gli operatori orientati ai bit sono AND, OR, XOR e NOT ed hanno, rispettivamente, i seguenti simboli: &, |, ^, ~. Oltre a questi ci sono gli operatori di shift << e >>. Gli effetti dei primi è riassunto nella seguente tabella:

X	Y	X & Y	X | Y	X ^ Y	~X
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Gli operatori di shift invece spostano i bit verso sinistra (<<) o verso destra (>>) del numero di posizioni specificate. Ad esempio se abbiamo un byte b col seguente valore in binario, 11011101 e gli applichiamo l'operatore << così: b << 2, otteniamo il valore 01110100.
Come si può vedere i primi due bit sono stati "buttati fuori" e gli altri spostati di due posizioni verso sinistra riempendo con zeri i bit di destra mancanti.
L'operatore >> è il complementare di <<.

Come si può vedere, tale classe che sarà utilizzata sia dall'*Encoder* sia dal *Decoder*, ha un solo metodo statico. Tale metodo si occupa di criptare sia la lunghezza del testo, sia il testo vero e proprio prima di inserirlo nell'immagine nel processo di codifica.
Nel processo di decodifica si occuperà di decriptare i byte estratti dallo stego file. Questo aggiungerà un'ulteriore sicurezza in quanto i bit che verranno inseriti nell'immagine non saranno direttamente quelli costituenti il nostro messaggio. A tale scopo viene utilizzata la classe *PasswordDeriveBytes* che fa parte del namespace *System.Security. Cryptography.* Questa classe riceve nel costruttore la stringa rappresentante la nostra password e un array di byte (ho semplicemente utilizzato la mia data di nascita come array di byte). Come si può notare se non viene inserita nessuna password ne viene usata una di default *("ioProgrammo")* per garantire che il messaggio venga comunque criptato prima di essere inserito nell'immagine.
Il metodo *GetBytes(128)* restituisce un array di 128 byte pseudocasuali.
Tale array verrà poi usato ciclicamente per "xorare" i byte costituenti il nostro messaggio con i byte restituiti da *GetByte*.

GLOSSARIO

GETPIXEL E SETPIXEL
Tali metodi fanno parte della classe *Bitmap* che si trova nel namespace *System.Drawing*. Mediante questi metodi è possibile leggere e settare i pixel di un'immagine utilizzando codice managed.

STEGANALISTA
Lo steganalista è colui che si occupa di steganalisi. Quest'ultima è un'arte tramite la quale, usando varie tecniche, si cerca di risalire al messaggio nascosto all'interno del cover

ESTRAZIONE DI UN MESSAGGIO SEGRETO DA UN'IMMAGINE

Se per la classe *PicEncoder* il metodo più interessante è *Encode*, per la classe *PicDecoder* il metodo più importante è, ovviamente, *Decode*. Tale metodo riceve in ingresso:

- una stringa rappresentante il percorso in cui si trova l'immagine da decodificare
- una stringa rappresentante la password da utilizzare per decriptare la lunghezza del messaggio e il messaggio vero e proprio estratti dall'immagine

Vediamone il codice completo:

```
public void Decode(string stegoFile, string pw) {
  bitmap = new Bitmap(stegoFile);
  maxWidth = bitmap.Width * 3;
  password = pw
  LockBitmap();
  byte[] testoLength = Extract(lunghezzaTesto);
  Stream length = Crypt.EncryptDecrypt(new
                  MemoryStream(testoLength), password);
  length.Read(testoLength, 0, 4);
  length.Close();
  int streamLength = BitConverter.ToInt32(
                                  testoLength, 0) * 8;
  int numByteMax = (bitmap.Height * bitmap.Width
                  * 3) - (lunghezzaTesto / 8);
  if(streamLength < 1 || streamLength > numByteMax)
    throw new Exception("In questa immagine non c'è
         nascosto alcun testo o la password è errata");
  byte[] testo = Extract(streamLength);
  message = Crypt.EncryptDecrypt(new
                    MemoryStream(testo), password);
  UnlockBitmap();
}
```

LockBitmap e *UnlockBitmap* li abbiamo già visti per la classe *PicEncoder*, come pure la classe *Crypt* che in questo caso si occupa di decriptare il testo. La cosa interessante da notare in questa classe è, oltre al metodo *Extract* che vedremo tra poco, il controllo che viene fatto sulla variabile *streamLength* che rappresenta la lunghezza del testo nascosto e che è stata estratta dall'immagine. Tale controllo è necessario in quanto se come stego file viene selezionata un'immagine che non contiene alcun testo nascosto o se l'immagine contiene un testo nascosto ma la password inserita è errata allora in *streamLength* ci può andare a finire qualsiasi valore causando un potenziale overflow. Il metodo *Extract* è quello che si occupa dell'estrazione dei byte rappresentanti la lunghezza del messaggio e del messaggio vero e proprio. Vediamone l'implementazione:

```
private byte[] Extract(int numBit) {
  int numBytes = numBit / 8;
  byte[] output = new byte[numBytes];
  int offset = maxWidth % 4;
  for(int i = 0; i < numBytes; i++) {
    byte current = 0;
    for(int j = 0; j < 8; j++) {
      byte b = *pointer;
```

```
            byte bit = Bit.Extract(b, 7);
            Bit.Replace(ref current, j, bit);
            pointer++;
            contatore++;
            if(offset != 0 && contatore == maxWidth){
              pointer += 4 - offset;
              contatore = 0;} }
    output[i] = current; }
  return output;
}
```

Come si può vedere, tale metodo estrae dall'immagine il numero di bit passato come parametro e lo restituisce sotto forma di array di byte. Sia il metodo *Replace,* della classe *PicEncoder,* sia il metodo *Extract,* della classe *PicDecoder,* fanno uso dei due metodi statici della classe *Bit.* Tali metodi svolgono il lavoro "sporco" di tutta l'applicazione, quello di basso livello. Cioè si occupano di rimpiazzare o estrarre i bit dei byte dell'immagine che costituiscono il nostro messaggio.

LA CLASSE BIT

La classe Bit, come si è visto, viene usata dalle classi PicEncoder e PicDecoder. Tale classe è formata da due metodi statici. Il primo, Replace, ha il seguente codice:

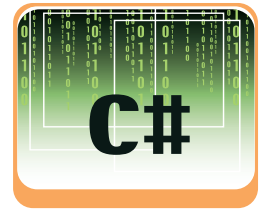
```
public static void Replace(ref byte b, int pos, byte valore) {
b = (byte) ((value == 1) ? b | (1 << (7-pos)) : b &
                                        ~(1 << (7-pos))); }
```

Questo metodo sostituisce il bit in posizione pos di un byte con un altro bit (rappresentato dal byte valore). Per chi non ha dimestichezza con gli operatori orientati ai bit si rifaccia al box laterale. Il secondo metodo, Extract, lo abbiamo così implementato:

```
public static byte Extract(byte b, int pos) {
return (byte) ((b & (1 << (7-pos))) >> (7-pos)); }
```

Come è ovvio aspettarsi, tale metodo estrae dal byte b il bit che si trova in posizione pos.
Nota: pos è un intero compreso tra 0 e 7 inclusi. 0 rappresenta il primo bit di un byte mentre 7 l'ottavo.

APPLICAZIONI DELLA STEGANOGRAFIA

Il campo in cui viene maggiormente utilizzata la steganografia è quello dei diritti d'autore. Pensiamo ad esempio alle immagini prodotte utilizzando dei programmi di grafica. Chi immaginerebbe mai che all'interno di queste immagini ci sia la "firma", comunemente detta *watermark,* del programma con il quale è stato prodotto? Eppure per le applicazioni più serie è proprio così! Questo viene fatto per preservare il diritto d'autore il quale può dimostrare in qualsiasi momento che quella specifica immagine è stata prodotta utilizzando il proprio programma risalendo al *watermark* inserito all'interno della stessa.

CONCLUSIONI

In questo articolo abbiamo visto cosa è la steganografia ed implementato una delle tecniche steganografiche moderne per scrivere una nostra applicazione. Abbiamo cercato di rendere comunque l'argomento quanto più semplice e pratico possibile, cercando di non entrare troppo nel dettaglio tecnico. Ad esempio per la codifica abbiamo inserito i bit del nostro messaggio un byte dietro l'altro, anche se prima di farlo abbiamo comunque criptato i byte da nascondere ottenendo una maggiore sicurezza. In un'applicazione per il mondo reale sarebbe più sicuro usare delle permutazioni pseudocasuali per "scegliere" la posizione del byte in cui nascondere il nostro bit pseudocasualmente volta per volta, ma questo avrebbe complicato ulteriormente l'argomento.
Ultime 2 note:

Fig. 4: *Class Diagram steganografato, con all'interno il sottotitolo di questo articolo*

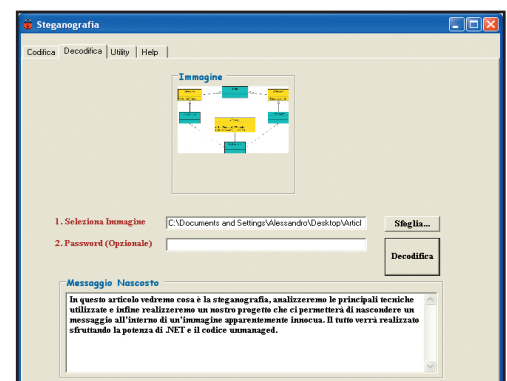



Fig. 5: *L'applicazione stego in fase di decodifica*

- Il nostro progetto di steganografia è stato sviluppato come dll, quindi è effettivamente indipendente dalla particolare applicazione .NET che lo utilizzerà.
- Per i più scettici, in **Figura 4** è riproposto il *Class Diagram* del progetto. Notate qualche differenza a livello ottico con la **Figura 3**?
  Eppure nell'immagine di **Figura 4** è stato inserito il sottotitolo di questo articolo. Chi *"non si fida"* potrà usare il programma che si trova nel CD per decodificare l'immagine *Fig4.bmp* anch'essa nel CD, ed ottenere il risultato di **Figura 5**.

In ultima analisi, si può vedere il cambiamento subito dall'immagine: è impercettibile.
Potenza della steganografia!

Alessandro Lacava

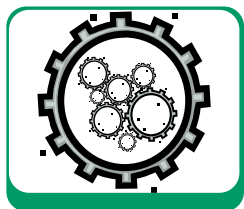
- **INFORMATION HIDING TECHNIQUES FOR STEGANOGRAPHY AND DIGITAL WATERMARKING** ***Stefan Katzenbeisser, Fabien A. P. Petitcolas*** **(Artech House Publishers)**

- **INVESTIGATOR'S GUIDE TO STEGANOGRAPHY** ***Greg Kipper*** **(Auerbach Publications)**

Impariamo a usare le API per la gestione della rete in VB

A spasso per Gruppi e Domini in VB

In questo articolo, utilizzeremo le funzioni di VB per la rete, "sfoglieremo" il network aziendale, e infine realizzeremo un progetto completo che ci aiuti nell'ottenere informazioni sullo stato della rete

CD WEB
Netapivb.zip
cdrom.ioprogrammo.it

Se avete avuto occasione di utilizzare Visual Basic per effettuare operazioni legate alla rete, avrete già scoperto la libreria *IPHlpApi.dll*. Sappiate che per la soluzione di molti problemi legati al networking in Visual Basic, ne esiste una seconda chiamata *NetApi-32.dll*.

Il progetto che stiamo per analizzare è stato realizzato in Visual Basic 6 e testato su un notebook Windows XP Professional, avviato all'interno di una rete aziendale. Il suo scopo è quello di mostrare l'elenco dei domini, dei PC e dei server presenti all'interno di ogni dominio o workgroup e l'elenco degli utenti registrati.

LE FUNZIONI PRINCIPALI

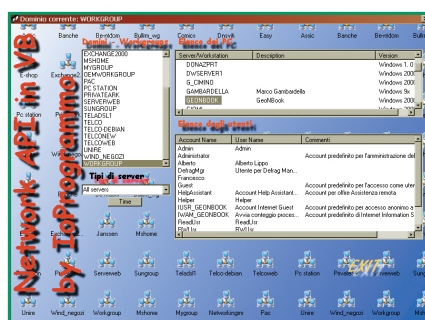
Una delle prime azioni compiute dal programma è la ricerca dei domini e dei workgroup rilevabili all'interno della propria rete. Questa operazione ha inizio attraverso la chiamata alla procedura *ListaDomain()*.

La funzione più importante all'interno di tale procedura è:

```
ServerList = EnumServer(SV_TYPE_DOMAIN_ENUM)
```

Tramite la quale otterremo l'elenco cercato. La funzione *EnumServer()* è dichiarata all'interno del modulo .BAS, allegato al progetto, nel modo seguente:

COSA FA LA NOSTRA APPLICAZIONE

Nel pannello di sinistra vengono elencati, in ordine alfabetico, tutti i domini e workgroup presenti all'interno della rete, mentre nei due pannelli a destra sono mostrati rispettivamente i PC ed i server del dominio/workgroup selezionato nel primo pannello ed alcune informazioni sugli utenti registrati. Sotto il pannello relativo all'elenco dei domini e dei workgroup è presente un controllo di tipo Combobox all'interno del quale sono stati inseriti una serie di item che consentono, se selezionati, di filtrare l'elenco dei server e dei PC mostrati, in base al loro "ruolo" all'interno della rete. Con questo termine identifichiamo alcuni particolari servizi installati: possiamo mostrare tutti i PC Terminal Server oppure tutti i Domain Controller del dominio o, ancora, le macchine SQL Server, ecc. Il valore di default è *All servers* ossia, se non diversamente specificato, il secondo pannello mostra l'intero

L'interfaccia principale del programma

elenco di macchine presenti nel workgroup o dominio selezionato. Dal punto di vista del codice implementato, il progetto è costituito soltanto da una form principale e da un modulo all'interno del quale sono state dichiarate le strutture, le costanti e le funzioni utilizzate all'interno del programma. Il funzionamento del progetto, da un punto di vista logico, è molto semplice:

1) **All'avvio, il programma richiama alcune funzioni che gli consentono di ottenere la lista dei domini e dei workgroup della rete.**
2) **Successivamente, viene rilevato il nome del dominio/workgroup di appartenenza del computer attualmente in uso.**
3) **A questo punto, sfruttando l'informazione precedente, viene "popolato" il secondo pannello, quello relativo alla lista dei server e dei PC presenti all'interno del dominio/workgroup considerato.**
4) **Con un doppio click su un qualunque item di quest'ultimo pannello, viene infine popolato quello relativo agli utenti (che riporta solo parte delle informazioni ricavate per ognuno di essi).**
5) **Con un doppio click del mouse su un qualunque item del pannello degli utenti viene mostrata a video una finestra contenente informazioni maggiormente dettagliate sull'utente considerato.**

```
Public Function EnumServer(ServerType As Long,
                Optional Domain As String) As ListaServer
```

Il primo parametro, *ServerType*, specifica il tipo di "filtro" da applicare all'elenco dei PC da mostrare a video.

LE AZIONI COMPIUTE ALL'AVVIO

All'avvio del programma, con l'attivazione dell'evento *Form_Load()* di *frmBrowseNet*, sono compiute diverse operazioni al fine di popolare i controlli nella form principale.

1 RIEMPIAMO IL COMBO CHE UTILIZZEREMO COME FILTRO

```
With Combo1
  .Items.Add(New VB6.ListBoxItem("All servers",
                                  SV_TYPE_ALL))
  ' Sistema operativo specifico
  .Items.Add(New VB6.ListBoxItem("NT/2000/XP/2003
              workstations or servers", SV_TYPE_NT))
  .Items.Add(New VB6.ListBoxItem("Windows 95/98/Me",
                                  SV_TYPE_WINDOWS))
  .Items.Add(New VB6.ListBoxItem("Windows for
                           Workgroups", SV_TYPE_WFW))
  ' .AddItem "Servers running Unix"
  ' .ItemData(.NewIndex) = SV_TYPE_SERVER_UNIX
  ' Funzioni specifiche
  .Items.Add(New VB6.ListBoxItem("Workstations
(servizio Workstation ON)", SV_TYPE_WORKSTATION))
  [..]
End With
```

Qui abbiamo riportato solo alcuni dei valori che possono essere filtrati. Chiaramente, utilizzeremo il secondo parametro per passare delle informazioni alla funzione che ci servirà per recuperare l'elenco dei domini o dei computer richiesto.

2 RECUPERIAMO I DOMINI E WORKGROUP

```
Sub ListaDomain()
   Dim Cont As Integer, xItem As ListItem
   Dim ServerList As ListaServer
   MousePointer = vbHourglass
   Me.SrvList.ListItems.Clear
   ' Domini/workgroup da elencare
     ServerList = EnumServer(
                             SV_TYPE_DOMAIN_ENUM)
   If ServerList.Init Then
      For Cont = 1 To UBound(ServerList.List)
         Me.DomainList.AddItem ServerList.List(
                                 Cont).ServerName
      Next
   End If
   MousePointer = vbDefault
End Sub
```

La funzione *ListaDomain()* recupera l'elenco dei domini e dei workgroup presenti all'interno della rete e per ogni elemento, aggiunge un Item al ComboBox relativo.

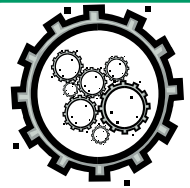
3 A QUALE DOMINIO APPARTIENE IL PC CHE ESEGUE L'APPLICAZIONE?

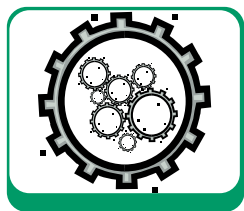
```
' Mostra l'elenco dei Domini/Workgroup
   ListaDomain
   MousePointer = vbHourglass
 ' Cerca il Dominio/Workgroup di appartenenza
                                       attuale...
   Dim DName As String
   DName = DomainName(Environ$(
                                "COMPUTERNAME"))
 ' ... ed evidenzialo all'interno della lista dei domini/
                                         workgroup
   Item = SendMessage(DomainList.hwnd,
         LB_FINDSTRINGEXACT, -1, ByVal DName)
   DomainList.ListIndex = Item
```

Questa funzione è molto semplice: non fa che recuperare le informazioni sul dominio di appartenenza del PC e lo evidenzia nella lista dei domini che abbiamo precedentemente caricato

4 QUALI COMPUTER SONO PRESENTI NEL DOMINIO?

```
Sub ListaComputer(DomainToBrowse As String)
   Dim Cont As Integer
   Dim xItem As ListItem
   Dim ServerList As ListaServer
   Dim VersionMajor As Long
   MousePointer = vbHourglass
   Me.SrvList.ListItems.Clear
   Me.UserList.ListItems.Clear
   ' Filtro sull'elenco dei PC
   ServerType = Combo1.ItemData(Combo1.ListIndex)
   ' Ottieni l'elenco dei PC
     ServerList = EnumServer(ServerType,
                                  DomainToBrowse)
   ' Se ci sono PC, allora mostra le principali
                                  caratteristiche
     If ServerList.Init Then
        For Cont = 1 To UBound(ServerList.List)
           Set xItem = Me.SrvList.ListItems.Add(
                  , , ServerList.List(Cont).ServerName)
           xItem.SubItems(1) = ServerList.List(
                                     Cont).Comment
           VersionMajor = (ServerList.List(Cont)
               .VerMajor And MAJOR_VERSION_MASK)
           ' ********************************
           ' If (ServerList.List(Cont).Type And
                         SV_TYPE_SQLSERVER) Then
           '  MsgBox ServerList.List(Cont).ServerName
           ' End If
           ' ********************************
           xItem.SubItems(2)=GetPlatformDescr(
             ServerList.List(Cont).PlatformId,Trim(Str(
               VersionMajor)) Str(ServerList.List(Cont)
                                        .VerMinor))
        Next
   End If
   Me.SrvList.Enabled = (Me.SrvList.ListItems.Count > 0)
   MousePointer = vbDefault
End Sub
```

Attraverso questa funzione recuperiamo l'elenco dei computer presenti in un dominio e riempiamo il controllo relativo.

L'ORARIO REMOTO

All'interno del nostro progetto è stato inserito anche un pulsante che mostra, qualora possibile, l'orario impostato sul PC remoto selezionato. Quest'ultima "feature" del programma offre la possibilità di conoscere un'ulteriore API di questa libreria, *NetremoteTOD()*, che potrebbe esservi utile per operazioni di sincronizzazioni tra client e server.

Il secondo parametro, *Domain*, determina il dominio o il workgroup di riferimento. Per essere più esaustivi, questa funzione ha due compiti ben precisi da svolgere, dipendenti proprio dai parametri passati:

- Ottenere l'elenco dei workgroup e dei domini presenti.
- Ottenere l'elenco delle macchine presenti all'interno di un workgroup o dominio.

I valori passati alla funzione, sono necessari al corretto funzionamento di un'API appartenente proprio alla libreria *NetApi32.dll*, denominata *NetEnumServer()*, sfruttata all'interno della funzione *EnumServer()*. La sintassi di questa funzione è la seguente (**Tabella 1**):

```
Declare Function NetServerEnum Lib "netapi32"
    (lpServer As Any, ByVal lLevel As Long, vBuffer As
  Any, lPreferedMaxLen As Long, lEntriesRead As Long,
    lTotalEntries As Long, ByVal lServerType As Long,
          ByVal sDomain$, vResume As Any) As Long
```

Al **passo 2** a presenza della funzione *ListaDomain()*, in realtà, è superflua, poiché tutto ciò che fa è semplicemente ricevere un array di strutture contenenti i dati inviati dalla funzione *NetServerEnum()*, dichiarata all'interno di *EnumServer()* e popolare di conseguenza la ListView opportuna. Tuttavia, per rendere maggiormente riutilizzabile il codice inserito, si è preferito separare queste due funzioni, facendo sì che, una chiamata alla funzione *EnumServer()*, con i parametri opportuni, ci restituisca "qualcosa" di riutilizzabile e svincolato da ciò che se ne vuol fare.

lpServer	sempre nullo
lLevel	specifica il livello relativo alle informazioni da ottenere. I valori possibili sono 100 e 101. A secondo del tipo di valore specificato, la funzione restituisce le informazioni all'interno di strutture simili, denominate rispettivamente *SERVER_INFO_100* e *SERVER_INFO_101*
vBuffer	puntatore ad un buffer che riceverà le informazioni. Esso viene allocato direttamente dal sistema e deve essere "svuotato" al termine attraverso una semplice chiamata alla *NetApiBufferFree()*, un'altra funzione della libreria *NetApi32.dll*
lPrefereMaxLen	specifica la massima lunghezza, espressa in byte, dei dati ricevuti.
lEntriesRead	puntatore ad un valore che identifica il numero di elementi attualmente conteggiati
lTotalEntries	puntatore ad un valore che identifica il numero totale di server e workstation rilevate
lServerType	filtro che consente di elencare soltanto alcune determinate categorie di macchine. Il valore di default, utilizzato dal programma attraverso l'uso degli item inseriti all'interno del controllo *Combo1*, è *SV_TYPE_ALL*. I valori possibili sono comunque visibili in testa alle dichiarazioni inserite all'interno del modulo *.BAS* allegato al progetto e possono essere utilizzati anche in combinazione tra loro. In particolare, utilizzando la costante *SV_TYPE_DOMAIN_ENUM*, otteniamo la lista dei workgroup e dei domini rilevati
sDomain	specifica il dominio o il workgroup per il quale elencare le informazioni richieste. Un valore nullo di questo parametro equivale al dominio/workgroup corrente. Tale stringa, affinché sia correttamente interpretata, va convertita in formato Unicode attraverso l'apposita funzione Visual Basic denominata *StrConv()*
vResume	sempre uguale a zero

Tabella 1: I parametri da passare a NetServerEnum

Come già detto più volte, la funzione *EnumServer()* consente di ottenere sia la lista dei workgroup e dei domini rilevati all'interno della rete sia quella relativa ai singoli PC ed ai server presenti. Nel primo caso, la funzione viene chiamata semplicemente utilizzando il primo parametro, impostato, come già visto, a *SV_TYPE_DOMAIN_ENUM*. Nel secondo caso, invece, è necessario specificare come primo parametro un valore che servirà da "filtro" per l'elenco dei computer da rilevare e come secondo parametro il workgroup o il dominio di riferimento.

L'ELENCO DEI COMPUTER

Nel caso volessimo ottenere l'elenco dei computer presenti nel dominio nel workgroup, non faremo altro che fare riferimento ancora una volta alla *EnumServer()* tuttavia specificheremo come primo parametro non più *SV_TYPE_DOMAIN_ENUM* che ci restituiva l'elenco dei domini, ma un nome di dominio più un filtro, che ci restituisce tutti i computer appartenenti ad un dato dominio e corrispondenti al filtro dato. Prima di passare alla successiva sezione, è bene dare un'occhiata alla porzione di codice commentato e delimitato dagli asterischi (*) nel **passo 4**.

Queste poche righe ci consentono di ottenere alcune informazioni più precise sulle macchine ritornate dalla *NetEnumServer()*. Infatti, attraverso delle operazioni di *AND* logico, possiamo verificare se una determinata macchina è, ad esempio, un *Primary Domain Controller* (PDC) ed, allo stesso tempo, costituisce anche un SQL Server. Nell'esempio, si è considerato solo quest'ultima ipotesi, ma possono essere combinati questi "flag" in maniera tale da poterli utilizzare in maniera più precisa. Inoltre, si osservi anche che, per motivi di semplicità, il filtro selezionabile è mutuamente esclusivo rispetto ai restanti, nel senso che, l'utilizzo di un "unico" item della *Combo1*, ci impedisce di applicare combinazioni di filtri. Questa osservazione può essere uno dei punti da migliorare all'interno del progetto.

LA STRUTTURA SERVERINFO

Avrete notato che l'elenco dei PC restituito da *EnumServer(…)* è stato inserito in una variabile *ServerList* di tipo *ServerInfo*. È importante comprendere come è stata concepita questa struttura:

```
Type ListaServer
    Init As Boolean
    LastErr As Long
    List() As ServerInfo
End Type
```

In particolare, la struttura *ServerInfo* è così dichiarata:

```
Type ServerInfo
    PlatformId As Long
    ServerName As String
    Type As Long
    VerMajor As Long
    VerMinor As Long
    Comment As String
    Platform As String
    ServerType As Integer
    LanGroup As String
    LanRoot As String
End Type
```

Non è complicato intuire a cosa si riferiscano i vari item di *ServerInfo*. In particolare, per quanto concerne l'utilizzo all'interno del programma, abbiamo:

- **PlatformId**: Identifica l'information level da utilizzare per le informazioni;
- **ServerName**: Nome del server;
- **Type**: Determina il tipo di servizio installato sulla macchina, ad esempio se si tratta di un server *Novell* o di un *Primary Domain Controller*, ecc;
- **VerMajor**: Rappresenta il *Major Release Version Number* del sistema operativo;
- **VerMinor**: Rappresenta il *Minor Release Version Number* del sistema operativo;
- **Comment**: Stringa che rappresenta un commento (in formato Unicode) sul server in oggetto. Può essere anche nulla;

INFORMAZIONI SUGLI UTENTI

La porzione di codice relativa al rilevamento delle informazioni sugli utenti registrati è a carico di un'altra funzione appartenente alla libreria *NetApi32.dll* denominata *NetUserEnum()*. La funzione suddetta è dichiarata in questo modo:

```
Declare Function NetUserEnum Lib "netapi32"
  (lpServer As Any, ByVal level As Long, ByVal Filter As
    Long, lpBuffer As Long, ByVal PrefMaxLen As Long,
         lpEntriesRead As Long, lpTotalEntries As Long,
                    lpResumeHandle As Long) As Long
```

Se la funzione viene chiamata con successo, il valore di ritorno è pari alla costante NERR_SUCCESS. Quando l'utente fa doppio click su un qualunque PC della rete, vengono rilevati, se possibile, tutti gli utenti registrati su quella macchina, con le informazioni relative alla descrizione ed agli eventuali commenti inseriti per ognuno di essi. Qualora si desiderasse ricevere maggiori dettagli su un particolare utente è possibile fare un doppio click sull'item voluto. Questa operazione genera l'apertura di una nuova finestra, all'interno della quale sono mostrati tutti i dettagli ritornati dalla funzione precedente. Dal punto di vista del codice implementato, l'operazione di doppio clic su un particolare PC della lista, richiama la funzione *ListaAccount()* che, analogamente a quanto vista prima per i domini ed i workgroup, richiama la funzione *EnumUsers()*. Anche qui valgono le considerazioni fatte precedentemente, nel senso che la *EnumUsers* si serve dell'API *NetUserEnum()* per ottenere informazioni circa gli utenti registrati all'interno di un computer. Tutti questi dettagli sono memorizzati all'interno di una struttura pubblica denominata *ListaUtenti* che viene richiamata dalla form *frmUserDetails*. Possiamo così visualizzare tutti i dettagli in una finestra separata, quella che potete ammirare in **Figura 2**.

Francesco Lippo

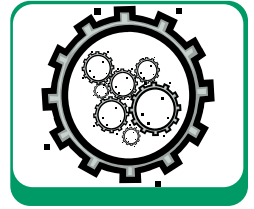



Fig. 2: La finestra con i dettagli sugli utenti

lpServer	puntatore ad una string ache identifica il nome DNS o NetBIOS di un server. Nel caso questo valore risulta essere nullo, la funzione considera come valido il PC correntemente in uso. In Windows NT la stringa deve essere formattata nella forma \\<*NomeServer*>.
level	analogamente a quanto visto per la funzione *NetServerEnum()*, identifica il livello informativo richiesto. I valori possibili sono 0, 1, 2, 3, 10, 11, 20, 23 (quest'ultimo non supportato su Windows 2000 e Windows NT). Il valore utilizzato all'interno del progetto è 3. Per la descrizione sugli elementi delle possibili strutture informative ritornate dalla funzione e, soprattutto, sull'uso e sul significato del vari item, si rimanda alla documentazione disponibile su MSDN.
Filter	sempre per impostare correttamente un filtro sugli utenti che la funzione dovrà ritornare. Un valore di Filter pari a zero indica alla funzione di ritornare informazioni su tutti gli utenti.
lpBuffer	puntatore ad un buffer che conterrà i dati sugli utenti.
PrefMaxLen	dimensione in byte massima relativa ai dati che la funzione dovrà ritornare. Un valore pari a MAX_PREFERRED_LENGTH fa sì che NetUserEnum() allochi la quantità di memoria necessaria ai dati da ritornare.
lpEntriesRead	puntatore ad un valore che indica il numero di elementi attualmente rilevati.
lpTotalEntries	puntatore ad un valore numerico che indica il totale degli item rilevati.
lpResumeHandle	puntatore ad un valore che consente di continuare la ricerca partendo da una precedente. Inizialmente questo valore è zero.

Tabella 2: I parametri da passare a NetUserEnum

Windows Management Istrumentation: potente e semplice

Controllo completo del sistema

Come si possono ottenere le informazioni sulle componenti hardware e software del vostro computer? In questo articolo impareremo ad utilizzare gli strumenti per la diagnostica e l'amministrazione di Windows

I sistemi operativi più recenti prodotti da Microsoft, ossia Windows 2000/XP/2003, supportano pienamente una tecnologia poco nota ma quanto mai potente: *WMI (Windows Management Instrumentation,* in italiano Strumentazione Gestione Windows). L'articolo ha la finalità di illustrare i concetti-chiave necessari per muovere i primi passi con questo illustre sconosciuto. Grazie a WMI le risorse di Windows possono essere facilmente rilevate, gestite, configurate o eliminate. Vi propongo qualche esempio pratico in cui la conoscenza delle classi WMI può tornare utile:

- **Informazioni hardware/software**: è possibile ottenere informazioni dettagliate relative ad ogni dispositivo presente sul computer.

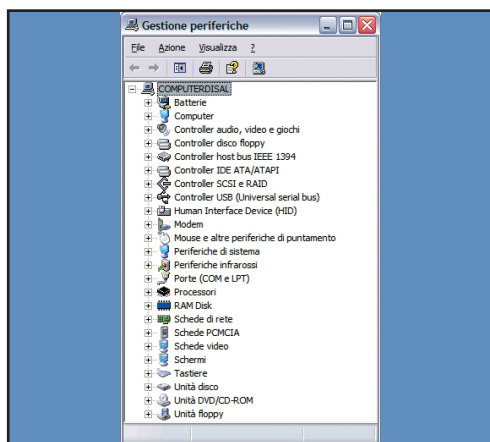



Fig. 1: *Il nostro progetto fornirà più informazioni di Gestione Periferiche!*

- **Amministrazione:** si possono aggiungere, modificare o rimuovere dispositivi o entità di qualsiasi tipo. Quante volte avreste voluto gestire utenti, processi, dischi, IIS, MSSQL Server o più in generale configurare Windows via codice? WMI isola lo sviluppatore dalla complessità delle funzioni a basso livello esposte dall'API del sistema operativo.
- **Analisi delle prestazioni**: gli indicatori delle prestazioni, per esempio l'utilizzo delle risorse di rete o della CPU, sono immediatamente accessibili tramite la strumentazione di gestione.

Vedremo come utilizzare WMI scrivendo query in un linguaggio simile a SQL.

LE BASI

L'architettura WMI (**Figura 2**) è costituita da 3 livelli: client, CIM e provider. Il client rappresenta il programma che sviluppiamo per amministrare Windows oppure per ottenere informazioni sul sistema. Gli oggetti CIM sono astrazioni di entità concrete: stampanti, dischi rigidi, reti, utenti, parametri di configurazione, etc. Un provider può essere visto come un intermediario tra l'oggetto fisico e la sua rappresentazione CIM. Windows è dotato di un'ampia gamma di provider, per i nostri scopi risulterà fondamentale il provider Win32. Tra il client ed il livello CIM esiste un'interfaccia COM/DCOM, dunque Visual Basic, Delphi, C++, VBScript sono solo alcuni dei linguaggi che consentono l'interazione con la Strumentazione di gestione.

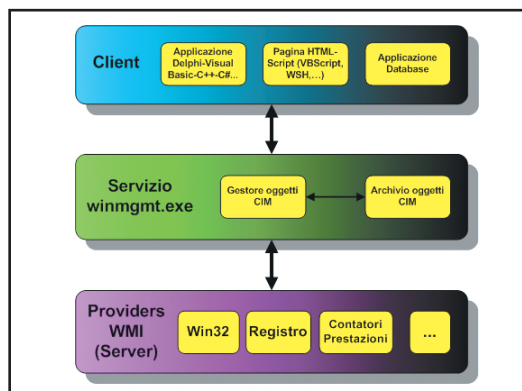



Fig. 2: *Una semplificazione dell'architettura (3-tier) WMI*

J2SE

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

IL LINGUAGGIO WQL

Per non complicare le cose gli esempi proposti nell'articolo si basano sull'accoppiata C#-WQL. La scelta è legata alla presenza di una serie completa di classi WMI messe a nostra disposizione dal .NET Framework. Il *WQL (WBEM* o *WMI Query Language)* è una variante del più noto SQL progettata appositamente per la gestione tramite WMI. A differenza del linguaggio SQL il WQL opera con classi e non con tabelle, inoltre non è possibile inserire, eliminare o modificare classi WMI utilizzando query WQL. Se volete esplorare le classi WMI presenti sul vostro computer potete ricorrere a *wbemtest* (**Figura 3**), tra le varie funzioni il programma prevede un ambiente per testare query WQL. Vi propongo una query banale ma molto interessante:

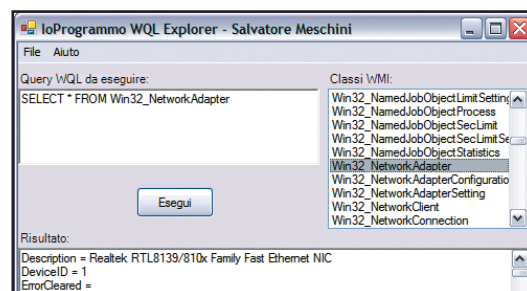



Fig. 3: **Eseguendo ..\system32\wbem\wbemtest.exe è possibile esplorare le classi WMI senza scriverecodice. Se non vi basta potete scaricare il pacchetto "WMI Microsoft Tools"**

```
SELECT Caption, FreeSpace, Size, VolumeName FROM
                win32_LogicalDisk WHERE caption='c:'
```

Il codice, tradotto da WQL in italiano, corrisponde alla richiesta *"SELEZIONA Lettera, Spaziolibero, Dimensione, Etichetta DA win32_LogicalDisk DOVE Lettera='c:'"*. Il risultato della richiesta è il seguente:

```
Caption = C:
FreeSpace = 11813273600
Size = 19510329344
VolumeName = SalvatoreMeschini
```

Pongo l'accento sulla possibilità di interrogare il sistema con query simili per ricavare tutte le informazioni di cui avete bisogno. Quando non si ha la necessità di filtrare una query mediante proprietà specifiche si può usare l'asterisco:

```
SELECT * FROM Win32_CDROMDrive
```

IL NOSTRO PROGRAMMA

L'esempio descritto nel seguito dell'articolo (**Figura 4**) mostra come creare una lista di oggetti di gestione e come eseguire query WQL. Le classi WMI non si limitano alla funzione di reperimento dei dati ma consentono una gestione completa del sistema operativo e delle applicazioni supportate. Il .NET Framework SDK contiene una miriade di utility troppo spesso ignorate dai programmatori. Il programma *mgmtclassgen.exe* è tra questi ed agevola notevolmente la scrittura di programmi che accedono a istanze delle classi WMI. Digitando *mgmtclassGen Win32_Logicaldisk /L CS /N root\cimv2 /P gestionedisco.cs* dal prompt del DOS otterrete una classe gestita in codice C#. Se sostituite CS con VB nel comando precedente verrà generata una classe gestita in codice VB.NET. Grazie a queste classi gestite le istanze WMI sono richiamabili in modo immediato:

Fig. 4: **WQL Explorer è il progetto che intendiamo realizzare. Attenzione: alcune query richiedono molto tempo!**

```
using System;
using System;
using System.Management;
using ROOT.CIMV2.Win32; //  GestioneDisco.cs
public class Sal
{
  public static void Main()
  {
    // Istanza di LogicalDisk (definita in GestioneDisco.cs)
                          in questo caso non uso codice WQL!
    Logicaldisk Disco = new Logicaldisk(new
        ManagementPath("win32_LogicalDisk.DeviceId=
                                                \"c:\""));
    // Mostra le informazioni
    Console.WriteLine("Su {0} hai ancora {1} bytes
                 liberi!\nPremi Invio per continuare...",
                        Disco.Caption, Disco.FreeSpace);
    Console.Read();
  }
}
```

ENUMERAZIONI ASINCRONE

In alcuni casi una query WQL restituisce una quantità ingente di dati richiedendo un tempo non indifferente. Per evitare che l'interfaccia utente risulti bloccata durante una richiesta lunga, è importante operare in modalità asincrona. Si potrebbe ricorrere ad un *thread* distinto ma ciò comporta diversi problemi di sincronizzazione. Inoltrandosi nei meandri del .NET Framework ci si imbatte nel-

SUL WEB

http://msdn.microsoft.com/library
Cercate i riferimenti a WMI nel sito Microsoft dedicato ai programmatori. Vi suggerisco di scaricare il pacchetto "*WMI Microsoft Tools*".

http://forum.ioprogrammo.net
La community di ioProgrammo, un buon punto di partenza per trovare la soluzione ai vostri problemi.

http://smeschini.altervista.org
Codice sorgente, trucchi ed altre risorse utili.

http://www.sharpdevelop.net
Un ottimo ambiente di sviluppo da associare al .NET Framework SDK.

la classe *ManagementOperationObserver*. Tale classe è stata concepita per gestire operazioni asincrone senza introdurre troppe complicazioni. *ManagementOperationObserver* espone quattro eventi pubblici, solitamente si preferisce rispondere a *Completed* oppure ad *ObjectReady*. Il primo evento viene generato quando un'operazione giunge al termine, il secondo viene generato quando un oggetto di gestione risulta disponibile. Dopo aver creato una nuova istanza con *ManagementOperationObserver osservatore = new ManagementOperationObserver();* bisogna assegnare un gestore di evento all'osservatore:

```
osservatore.ObjectReady +=new
        ObjectReadyEventHandler(OggettoPronto);
```

A questo punto non resta che implementare la funzione *callback OggettoPronto* in maniera opportuna:

```
void OggettoPronto(object sender, ObjectReadyEventArgs e)
{
  // Qui viene aggiornata l'interfaccia utente
}
```

Vi lascio un esercizio "per casa" molto istruttivo: modificate il programma di esempio *WQLExplorer* aggiungendo la capacità di gestire le enumerazioni asincrone.

CONCLUSIONI

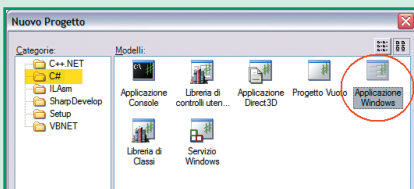
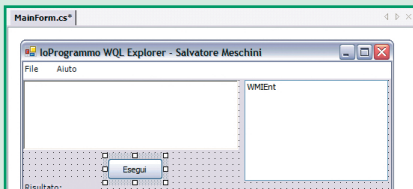
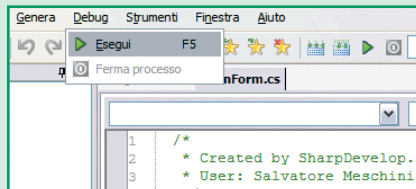
Parleremo ancora di infrastruttura WMI, creazione di WMI providers, invocazione di metodi, estensioni per Visual Studio, etc.
Confido nella vostra volontà di approfondire l'argomento.
Alla prossima puntata!

Salvatore Meschini

GLOSSARIO

SQL
***Structured Query Language* è il linguaggio più utilizzato da chi sviluppa applicazioni che interagiscono con basi di dati. Permette di estrarre, inserire, modificare, eliminare dati e metadati mediante "funzioni" dette *query*.**

LO SCHELETRO DEL PROGRAMMA

1 CREAZIONE DEL PROGETTO - Dopo aver avviato *#Develop* accediamo al menu *File->Nuovo->Progetto* oppure usiamo la combinazione di tasti *CTRL+SHIFT+N* per poi selezionare *"Applicazione Windows"* in linguaggio C#. Non dimentichiamo di scegliere un nome ed una cartella.

2 INTERFACCIA GRAFICA - I componenti utilizzati per creare l'interfaccia sono essenzialmente un Listbox, due *RichTextBox* ed un *Button*. Si trovano nella sezione *Windows Forms* della linguetta *Strumenti*. La disposizione finale non dovrebbe essere dissimile da quella mostrata in figura.

3 ESECUZIONE DEL PROGRAMMA - In *Develop* l'esecuzione di un progetto si avvia premendo il tasto *F5* come in Visual Studio. Prima di proseguire proviamo ad eseguire il progetto per escludere la presenza di errori. Sul CD allegato alla rivista trovate il sorgente completo di *WQLExplorer*.

4 SPAZIO DEI NOMI

```
using System;
using System.Windows.Forms;
using System.Text; // Classe StringBuilder
using System.Management;
                              // E' fondamentale!
namespace DefaultNamespace
{
  public class MainForm :
                    System.Windows.Forms.Form
  {
     // ...
  }
}
```

Dobbiamo aggiungere due namespace al nostro progetto perché abbiamo bisogno delle classi *WMI* e di una istanza di *StringBuilder*. Le direttiva *using* vanno inserite all'inizio del file *MainForm.cs*. *#Develop* supporta il completamento automatico del codice.

5 ENUMERAZIONE

```
ManagementClass WMIRoot = new
            ManagementClass(); // Radice
EnumerationOptions Opt = new
                  EnumerationOptions();
Opt.EnumerateDeep=true; // Scansione completa
WMIEnt.BeginUpdate(); // Evita lo sfarfallio
foreach(ManagementObject Cls in
            WMIRoot.GetSubclasses(Opt))
{ // Escludo tutte le classi diverse da Win32*
                              (es. CIM*)
 if (Cls["__Class"].ToString().StartsWith(
                              "Win32"))
WMIEnt.Items.Add(Cls["__Class"]);
                   // Aggiungo al Listbox }
WMIEnt.EndUpdate();
```

Otteniamo tutti gli oggetti di gestione con una scansione completa della radice *WMIRoot*. In *foreach* vengono filtrati i nomi degli oggetti. Quelli con prefisso *"Win32"* risultano presenti nel Listbox. *Cls["__Class"]* è il nome della classe.

6 QUERY WQL

```
SelectQuery MyQuery = new SelectQuery(
          WQLCodeBox.Text); // Codice WQL
ManagementObjectSearcher Results = new
     ManagementObjectSearcher(MyQuery);
StringBuilder RS = new StringBuilder();
               // Per accodare le stringhe
// Itera e aggiungi a RS (risultato)
foreach(ManagementObject R in Results.Get())
foreach(PropertyData Pta in R.Properties)
{ RS.Append(Pta.Name + " = " +
                       Pta.Value + "");
}
ResultBox.Text = RS.ToString();
                     // Mostra i risultati
```

Per eseguire una query WQL è necessario passare la stringa di codice WQL al costruttore di *SelectQuery*. L'istanza di *SelectQuery* è usata da *ManagementObjectSearcher*. Ogni coppia nome-proprietà viene aggiunta alla stringa risultato dal doppio ciclo *foreach*.

Utilizziamo JAI per costruire un programma Java di fotoritocco

Fotoritocco in Java

II parte

In questo articolo, costruiremo un menu, implementeremo una finestra di apertura file, caricheremo un'immagine dal file system e gli applicheremo interessanti effetti

Utilizza questo spazio per le tue annotazioni

Nel precedente numero di ioProgrammo abbiamo iniziato lo sviluppo di JAIPhoto (**Figura 1**), una semplice applicazione di manipolazione di immagini che utilizza le API JAI, *Java Advanced Imaging*. Abbiamo sviluppato l'interfaccia utente dell'applicazione, il caricamento delle immagini utilizzando i thread ed il ridimensionamento dell'immagine. In questa puntata si proseguirà nello sviluppo del programma, implementando alcuni effetti di manipolazione dell'immagine. Questi includeranno l'inversione dei colori, la sfocatura ed altro.

Fig. 1: **La finestra principale dell'applicazione JAIPhoto**

MENU PRINCIPALE

Con le API *Swing* è possibile associare una barra di menu ad ogni finestra implementata con la classe *JFrame*. Questa dispone infatti del metodo *setJMenuBar()* che permette di associare alla finestra un oggetto di tipo *JMenuBar*. Questa classe rappresenta una barra di menu, completa delle voci che la compongono. Ciascun menu è rappresentato da oggetti *JMenu*, mentre ogni voce di menu è rappresentato da oggetti *JMenuItem*. Inizialmente nel nostro esempio avremo due menu: *File* ed *Effetti*, come in **Figura 2**. Per creare un menu procedere come segue:

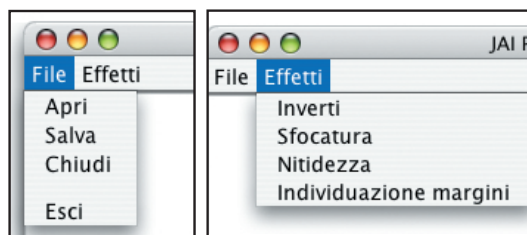



Fig. 2: **La struttura dei menu di JAIPhoto**

1 DICHIARAZIONE DEL METODO E CREAZIONE BARRA MENU - Il metodo createMenu() crea un oggetto JMenuBar con tutte le voci di menu necessarie. Viene creato un nuovo oggetto JMenuBar, che sarà la barra del menu dell'applicazione:

```
JMenuBar createMenu() {
        JMenuBar menuBar = new JMenuBar();
```

2 CREAZIONE DEL MENU FILE - questa operazione avviene con la creazione di una istanza della classe JMenu, il cui costruttore si aspetta il nome del menu:

```
JMenu file = new JMenu("File");
```

3 CREAZIONE DI TUTTE LE VOCI DEL MENU FILE - ciascuna voce di menu è realizzata con oggetti di tipo JMenuItem, che si aspettano nel costruttore la dicitura da utilizzare come menu. Inoltre, per collegare una azione ad una voce di menu, è necessario creare un oggetto ActionListener. Per ogni voce di menu viene infatti creata una classe anonima che implementa l'interfaccia ActionListener. Questa classe viene impostata come ascoltatore di

eventi del relativo elemento di menu. Ovviamente, per ciascuna voce di menu sarà presente un ActionListener diverso. Ciascuno conterrà la chiamata al metodo opportuno. Ad esempio, la voce di menu Apri chiamerà il metodo apri(), mentre Chiudi invocherà il metodo chiudi(). Fanno eccezione gli effetti, per cui il codice presente nel relativo ActionListener invocano già le classi che implementano gli effetti da applicare.

```
openMenuItem = new JMenuItem("Apri");
openMenuItem.addActionListener( new
ActionListener() {
   public void actionPerformed( ActionEvent e ) {
      apri(); }
});
saveMenuItem = new JMenuItem("Salva");
saveMenuItem.addActionListener( new
ActionListener() {
   public void actionPerformed( ActionEvent e ) {
      salva();}
});
closeMenuItem = new JMenuItem("Chiudi");
closeMenuItem.addActionListener( new
ActionListener() {
   public void actionPerformed( ActionEvent e ) {
      chiudi(); }
});
quitMenuItem = new JMenuItem("Esci");
quitMenuItem.addActionListener( new
ActionListener() {
   public void actionPerformed( ActionEvent e ) {
      System.exit(0);}
});
```

4 AGGIUNTA DELLE VOCI AL MENU, NELL'ORDINE DESIDERATO - questa operazione avviene utilizzando il metodo add(). Se si desidera raggruppare voci di menu è possibile utilizzare un separatore. Questo viene aggiunto con il metodo addSeaprator().

```
file.add( openMenuItem );
file.add( saveMenuItem );
file.add( closeMenuItem );
file.addSeparator();
file.add( quitMenuItem );
```

5 CREAZIONE DEL MENU EFFETTI E DELLE RELATIVE VOCI DI MENU - questo menu include gli effetti che saranno implementati nel corso dell'articolo:

```
JMenu effects = new JMenu("Effetti");
invertMenuItem = new JMenuItem("Inverti");
invertMenuItem.addActionListener( new
ActionListener() {
   public void actionPerformed( ActionEvent e ) {
      InvertEffect effect = new InvertEffect();
      effect.run(runner);
      image = runner.getImage(); } });
...
effects.add( invertMenuItem );
effects.add( blurMenuItem );
effects.add( unBlurMenuItem );
```

6 AGGIUNTA DEI MENU E RITORNO DEL METODO - una volta valorizzati i singoli menu, questi vengono aggiunti alla barra dei menu, che viene ritornata dal metodo createMenu():

```
   menuBar.add( file );
   menuBar.add( effects );
   return menuBar;
}
```

APRIRE UNA IMMAGINE

L'apertura di una immagine avviene invocando la voce di menu *Apri*, che richiama il metodo *apri()*. Questo metodo è implementato come segue:

```
JFileChooser fc = new JFileChooser();
int result = fc.showOpenDialog(frame);
if (result == JFileChooser.APPROVE_OPTION)
{
   String filename = fc.getSelectedFile().getAbsolutePath();
   load( filename );
}
```

Come si nota dal codice, viene utilizzato un oggetto *JFileChooser*. Questa classe, presente nelle API *Swing*, consente di presentare una finestra di selezione di file.
Questa tipologia di finestra viene aperta con il metodo *showOpenDialog()*. Il metodo si aspetta un parametro: il componente visuale a cui collegare la finestra di selezione file. In questo modo, la finestra principale risulterà bloccata fino a quando non verrà selezionato un file, oppure annullata l'operazione. È possibile utilizzare il metodo *getSelectedFile()* per conoscere il nome del file selezionato. In caso di *JAIPhoto*, questo viene passato al metodo *load()* per caricare l'immagine scelta.

Fig. 3: *L'immagine di prova con i colori invertiti*

INVERTIRE I COLORI DELL'IMMAGINE

Il primo effetto che verrà implementato è la classica inversione dei colori dell'immagine (**Figura 3**). Il codice che invoca questo effetto è descritto

CLASSE ANONIMA

Una classe anonima è una classe dichiarata nel corpo del codice senza specificarne il nome, subito dopo la chiamata all'operatore new. Se ne dichiara l'implementazione racchiudendo il relativo codice tra parentesi graffe, che sono posizionate dopo il nome della superclasse o dell'interfaccia da implementare.

APPROFONDIMENTI

SUN Microsystems mette a disposizione, sul proprio sito, una raccolta concisa di link a risorse di approfondimento sulla tecnologia JAI. Sono inclusi strumenti open source e commerciali, documentazione e suggerimenti compilati dagli sviluppatori. La pagina è disponibile all'indirizzo: http://java.sun.com/products/java-media/jai/utilities/jaiutils.html

JAI IN AZIONE

Volete vedere JAI in azione? Sono già diverse le aziende che stanno impiegando questa tecnologia nei loro prodotti. Ad esempio, Oracle la sta utilizzando per il suo database multimediale. Ma forse l'uso più famoso, accanto alle implementazioni per la medicina e per le foto digitali, è quello della NASA, che l'ha utilizzata per elaborare le immagini ottenute da Marte. I link si trovano alla pagina: http://java.sun.com/products/java-media/jai/inaction/

al punto 5 della procedura illustrata prima.
Le operazioni che compie sono: creazione dell'effetto, esecuzione e salvataggio dell'immagine modificata. Il codice è il seguente:

```
InvertEffect effect = new InvertEffect();
effect.run(runner);
image = runner.getImage();
```

Le operazioni sono dunque tutte racchiuse nella classe *InvertEffect*. In *JAIPhoto* ogni effetto è una implementazione dell'interfaccia *Effect*. Il codice dell'effetto è il seguente:

```
package it.bigatti.jaiphoto.effects;
import javax.media.jai.JAI;
import javax.media.jai.PlanarImage;
import it.bigatti.jaiphoto.utils.JAIOperation;
import it.bigatti.jaiphoto.utils.JAIRunner;
public class InvertEffect implements Effect {
  JAIRunner runner;
  public void run(JAIRunner jaiRunner) {
    runner = jaiRunner;
    runner.run( new JAIOperation() {
      public PlanarImage run() {
        PlanarImage image = JAI.create("invert",
                                    runner.getImage());
        return image; }
      public String getDescription() {
        return "invert"; }});}}
```

A parte le classiche istruzioni di importazione e la dichiarazione della classe, si nota la presenza del solo metodo *run()*. Questo si aspetta come parametro un oggetto *JAIRunner*. Come si ricorderà dal numero precedente, questo oggetto serve per eseguire operazioni JAI in modo asincrono. Nel metodo *run()* viene infatti creata una classe anonima che implementa l'interfaccia *JAIOperation*. Questa contiene la chiamata alla libreria JAI per invertire i colori. Nel metodo *run()* è infatti presente la chiamata a *JAI .create()* specificando l'operazione *"invert"*.

ALTRI EFFETTI E CONVOLUZIONE

Una operazione importante nell'elaborazione delle immagini è la "convoluzione". Questa operazione permette di applicare una matrice di elaborazione a ciascun punto dell'immagine. Grazie alla convoluzione è possibile ottenere diversi effetti, come la sfocatura, l'aumento del contrasto e l'identificazione dei margini. Esempi di questi tre effetti sono presenti nelle **Figure 4a, b, c**. Per applicare una sfocatura ad una immagine è necessario costruire una matrice di dati, detta *Kernel*, che dovrà essere utilizzata nell'operazione di convoluzione. Per creare l'effetto di sfocatura (*blur*), procedere come segue:

1 CREARE LA CLASSE DELL'EFFETTO CHE IMPLEMENTA L'INTERFACCIA EFFECT - tutti gli effetti sono posizionati nel package it.bigatti.jaiphoto.effects ed includono alcune classi chiave di JAIPhoto e JAI. L'inizio è una parte standard per ciascun effetto:

```
package it.bigatti.jaiphoto.effects;
import it.bigatti.jaiphoto.utils.JAIOperation;
import it.bigatti.jaiphoto.utils.JAIRunner;
import javax.media.jai.JAI;
import javax.media.jai.KernelJAI;
import javax.media.jai.PlanarImage;
public class BlurEffect implements Effect {
  JAIRunner runner;
  public void run(JAIRunner jaiRunner) {
    runner = jaiRunner;
    runner.run( new JAIOperation() {
      public PlanarImage run() {
```

2 CREAZIONE MATRICE DI CALCOLO DELLA CONVOLUZIONE - nel metodo run(), creare una matrice di dimensione 3 che contenga il valore 1/9. Questo valore porterà al risultato di ottenere una sfocatura. Altri valori comportano altri risultati. Si noti che nella matrice ogni elemento ha valore 1/9. Valorizzando la matrice con valori differenti per ciascuna cella produce risultati diversi, come si vedrà in seguito. La costruzione della matrice avviene semplicemente eseguendo due cicli annidati, che eseguono una iterazione per ciascuna dimensione della matrice.
Variando il valore della costante kernelSize è possibile ottenere una matrice di dimensioni differenti senza la necessità di eseguire altre variazioni al codice. Negli effetti implementati più avanti non verrà utilizzato questo sistema dinamico, ma la matrice avrà una dimensione fissa di tre per tre, e sarà inizializzata in modo statico:

```
int kernelSize = 3;
int c = 0;
float[] kernelMatrix = new float[kernelSize
                                 * kernelSize];
for(int k = 0; k < kernelSize; k++) {
  for(int i = 0; i < kernelSize; i++) {
    kernelMatrix[c] = 1.0f/9.0f;
    c++;
  }
}
```

3 CREARE L'OGGETTO KERNEL PER IL CALCOLO DELLA CONVOLUZIONE - l'oggetto KernelJAI viene creato specificando la dimensione della matrice ed il valore della stessa:

```
KernelJAI kernel = new KernelJAI(kernelSize,
                       kernelSize, kernelMatrix);
```

4 APPLICARE LA CONVOLUZIONE UTILIZZANDO IL KERNEL CREATO - per applicare l'operazione di convoluzione viene utilizzata un'altra versione del metodo JAI. create() che si aspetta un terzo parametro. Questo è proprio il kernel da utilizzare:

```
PlanarImage image = JAI.create("convolve",
                    runner.getImage(), kernel);
```

5 RITORNARE L'IMMAGINE AL SISTEMA - l'immagine ottenuta dall'elaborazione viene restituita al sistema per essere visualizzata:

```
    return image;
  }
```

6 CONCLUDERE LA CLASSE CON UNA DESCRIZIONE APPROPRIATA - come si ricorderà, ciascuna operazione JAIOperation dispone di una descrizione, che indica la natura dell'operazione. In questo caso la descrizione è "blur", termine inglese per sfocare:

```
  public String getDescription() {
     return "blur"; }
 });}}
```

EFFETTO NITIDEZZA

L'effetto è molto simile a quello di sfocatura, con la differenza che in questo caso la matrice delle operazioni è diversa. In questo caso la matrice non è costruita con un ciclo, ma inizializzando direttamente la variabile *kernelMatrix*. Questa viene riempita con valori tali da comportare la simulazione della nitidezza dell'immagine. Si noti che sia in questo effetto che nel precedente è stata utilizzata una matrice di tre elementi per tre. Un Kernel in realtà può supportare anche matrici più grandi. Ovviamente, più grande è la matrice maggiore sarà il tempo di elaborazione richiesto per applicare l'effetto. La classe che implementa la nitidezza si chiama *UnBlurEffect*. Come si nota osservando il codice presente sul CD allegato alla rivistao sul Web *www.ioprogrammo.it*, le operazioni di costruzione del Kernel e dell'applicazione dell'effetto sono le medesime rispetto all'effetto precedente. Il risultato dell'effetto è presente in **Figura 4b**.

INDIVIDUAZIONE DEI MARGINI

In inglese "*Edge Detection*", permette di estrarre da una immagine i contorni degli elementi presenti. Questo avviene mettendo in risalto le zone dove il colore cambia radicalmente. Dove il colore è uniforme, invece, la colorazione dell'immagine risultante assume tonalità vicine al nero. Il risultato di un effetto di individuazione dei margini è dunque una immagine molto scura, con i margini individuati dal linee più chiare. Per applicare un effetto di individuazione dei margini è necessario ricorrere ancora alla convoluzione. Questo effetto avviene infatti con una matrice molto simile a quello della nitidezza. In questo caso cambia solo il peso dell'elemento centrale, che diminuisce di uno. La matrice per l'individuazione dei margini è infatti:

```
float[] kernelMatrix = {
  0.0f,   -1.0f,    0.0f,
  -1.0f,   4.0f,   -1.0f,
  0.0f,   -1.0f,    0.0f   };
```

mentre quella per la nitidezza era:

```
float[] kernelMatrix = {
  0.0f,   -1.0f,    0.0f,
  -1.0f,   5.0f,   -1.0f,
  0.0f,   -1.0f,    0.0f   };
```

come si nota, la differenza è minima. Nonostante questo, c'è una enorme differenza tra il risultato delle due operazioni! L'individuazione dei margini è presente in **Figura 4c**. La si confronti con la **Figura 4b**, che è il risultato dell'effetto di nitidezza. Si noterà una sostanziale differenza. L'effetto di individuazione dei margini è implementato nella classe *EdgeDetectionEffect*.

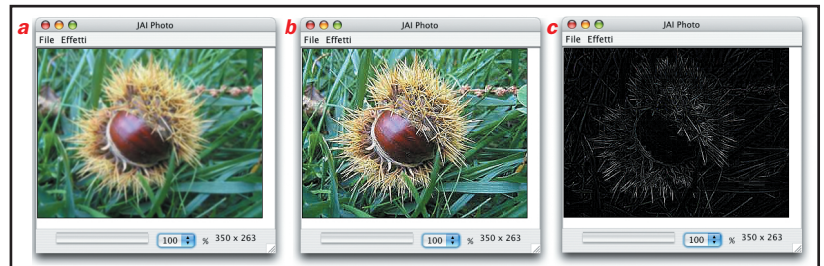



Fig. 4: *Un esempio degli effetti messi a disposizione da JaiPhoto*

CONCLUSIONI

È bene ricordare che JAI supporta i formati jpg, gif, TIFF e BMP. Tentare di aprire un file grafico non supportato produce un errore in fase di esecuzione, che però non viene presentato all'utente in modo molto amichevole. La gestione degli errori di JAIPhoto è infatti un'area ancora da sviluppare. Mancano alcuni effetti evoluti, come ad esempio la possibilità di eseguire una operazione di convoluzione su un Kernel di dimensioni e valori a piacere. Un'altra funzione presente nel menu ma non ancora implementata è il salvataggio dell'immagine modificata.

Massimiliano Bigatti

KERNEL

Il Kernel è un insieme di parametri che vengono considerati durante l'esecuzione di un effetto di convoluzione e che permettono di ottenere diversi effetti. Programmi come The Gimp consentono di specificare il proprio Kernel per eseguire una operazione di convoluzione assolutamente personalizzata.

CONVOLUZIONE

È un effetto molto diffuso che elabora l'immagine punto per punto, calcolando il valore del pixel di uscita in funzione del valore dei pixel contigui. Durante questa operazione utilizza un Kernel, che indica quale "peso" dare al valore del pixel che si sta elaborando ed a quelli vicini.

Progettiamo un'applicazione enterprise come ci suggerisce Sun

BluePrints: i pattern "naturali" di J2EE

II parte

Terminiamo la progettazione del nostro motore di ricerca, sfruttando i design pattern che vengono descritti nei Java BluePrints per J2EE. In questa II parte parleremo di DAO, Value Object e Session Façade

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
UML, J2EE

Software
BEA Weblogic Server 8.1 o application server equivalente. Un IDE come JBuilder X, Eclipse 3.0 o prodotto equivalente

Impegno

Tempo di realizzazione

Il mese scorso abbiamo iniziato la nostra esplorazione dei pattern di Java 2 Enterprise Edition. Abbiamo studiato come migliorare le nostre applicazioni basate su J2EE, utilizzando le soluzioni suggerite dai BluePrints della Sun. Abbiamo introdotto il pattern *Model View Controller (MVC)* per impostare l'architettura J2EE, ed abbiamo sviluppato un semplice motore di ricerca come esempio di codice. Ci siamo però limitati solo allo realizzazione dello "strato WEB", ovvero quello relativo a Servlet e JSP con cui abbiamo concretizzato rispettivamente il *Controller* e la *View* della nostra applicazione. Per raggiungere il nostro scopo nel modo migliore, abbiamo applicato i pattern *Service Locator, Business Delegate* e *View Helper.* In questo articolo completeremo l'applicazione con lo sviluppo dello "strato Business", ovvero quello relativo agli EJB, concretizzando così il componente che nell'MVC viene detto Model.
Per raggiungere il nostro scopo, in questo articolo saranno quindi trattati i seguenti pattern: *Data Access Object, Value Object,* e *Session Façade;* cercheremo di esplicitarne i vantaggi, realizzando contemporaneamente un interessante framework riutilizzabile.

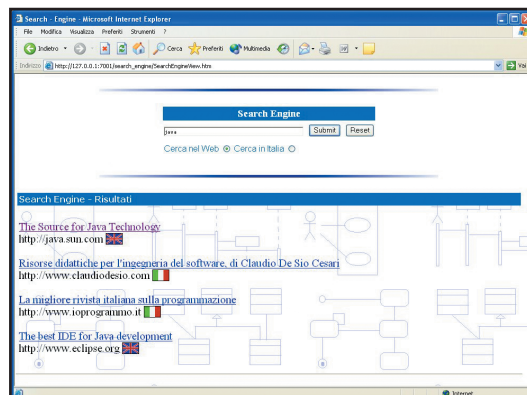



Fig. 1: Una screenshot del motore di ricerca che implementeremo

VIEW E CONTROLLER

Nel precedente articolo abbiamo introdotto il pattern che è alla base dell'architettura J2EE: il *Model View Controller* (o più semplicemente MVC). Questo pattern architetturale è caratterizzato da una certa complessità, ma garantisce grande efficienza.
La proprietà più evidente però, individuabile a partire dal nome, è quella di distinguere nettamente i tre componenti all'interno di un'applicazione, con un'esplicita assegnazione di responsabilità:

- la **View:** che implementa la logica di presentazione
- il **Controller:** che implementa la logica di controllo
- il **Model:** che implementa la logica di business, e quindi concretizza l'applicazione vera e propria.

Nel precedente articolo abbiamo creato lo strato WEB di un motore di ricerca basato su J2EE.
Con l'MVC abbiamo imposto un framework per la comunicazione tra le varie tecnologie, e ci ha quindi evitato di avventurarci in modelli comunicativi sperimentali. Ogni componente dell'applicazione aveva delle precise responsabilità, e quindi è stato semplice progettarne l'implementazione. Per esempio, le *servlet* fungono da Controller, implementando la logica di controllo dell'applicazione. In particolare, catturano le richieste del client e controllano la correttezza dei parametri. Successivamente, scelgono il giusto metodo del *Model* (che è la vera applicazione) da invocare, e reindirizzano i risultati ad una pagina JSP. Questa si deve occupare della logica di presentazione, e quindi il maggior sforzo implementativo risiederà nella creazione del layout della pagina di risposta al client. Una JSP infatti, è per sua natura costituita essenzialmente da tag per la formattazione della pagina. Il codice *scriptlet* dovrebbe essere limitato al minimo, se non escluso del tutto. Per il

nostro motore di ricerca il pattern *View Helper* ha aiutato la JSP a non contenere codice scriptlet. Il pattern *Service Locator* inoltre, ci ha permesso di evitare di scrivere codice complesso per accedere ad eventuali risorse remote come gli EJB. Inoltre ha permesso di migliorare le performance dell'applicazione mediante un semplice ma efficace meccanismo di *caching*, per evitare di invocare metodi remoti quando non ce n'è bisogno. Infine il pattern *Business Delegate*, supporta la creazione una classe che incapsula la logica di accesso allo strato *Business*, offrendo una interfaccia "friendly" alle servlet.

IL MODEL

Nella mia attività di consulente, ho visto spesso codice dello strato Web (in servlet o addirittura in JSP), accedere direttamente alla base dati. Le ragioni per cui tali pratiche siano da evitare sono state già chiarite nel precedente articolo, con la descrizione dell'architettura J2EE ed il pattern MVC.

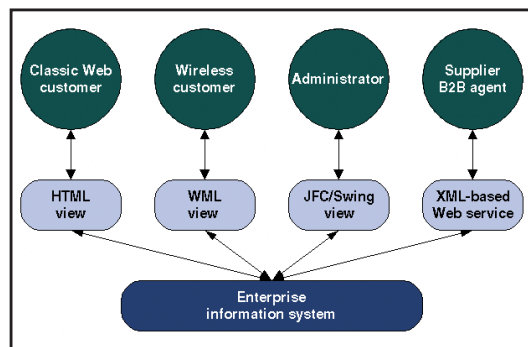



***Fig. 2:** Oggigiorno esistono varie tecnologie per accedere ad un'applicazione*

Se il *Model* è la vera applicazione con le sue regole di business, è solo lì che si deve accedere al database. Con questa distinzione, sarà possibile creare altre coppie *View-Controller* in modo tale da permettere, l'utilizzo dell'applicazione in contesti diversi. La tecnologia EJB ben si adatta a questo contesto. Infatti, tramite il protocollo *RMI-IIOP*, si potrà accedere all'applicazione in situazioni eterogenee. Questo è uno dei punti di forza di un'applicazione J2EE. Tuttavia, se accedessimo in maniera ripetuta con varie chiamate a granularità fine dallo strato Web allo strato *Business*, le performance dell'applicazione ne risentirebbero. Pensiamo per esempio ad un *entity bean* con interfaccia remota. Per settare un campo con un metodo *setXXX()*, è indispensabile invocare un metodo remoto! Nessuno vi impedirà mai di implementare *entity bean* con interfacce remote, ma è certo che non si potrebbe parlare di una felice scelta architetturale. Ecco perché le specifiche EJB 2.0 hanno introdotto le interfacce locali. In pratica, per accedere anche ad *entity bean* remoti, lo strato Web dovrà invocare metodi di un eventuale *session bean* con interfaccia remota, che si preoccuperà di invocare tramite interfaccia locale eventuali metodi sugli *entity bean*, con cui condivide lo spazio dello strato *Business*. Questo *session bean* non è altro che un'implementazione di un pattern fondamentale: il *Session Façade*.

SESSION FAÇADE

Il *Session Façade* rappresenta la versione per J2EE del pattern *GoF* denominato *Façade*. Esso si basa su di un'idea semplice e molto valida: implementare un session bean che fornisca un'interfaccia per le funzionalità dell'applicazione, mediante metodi remoti.

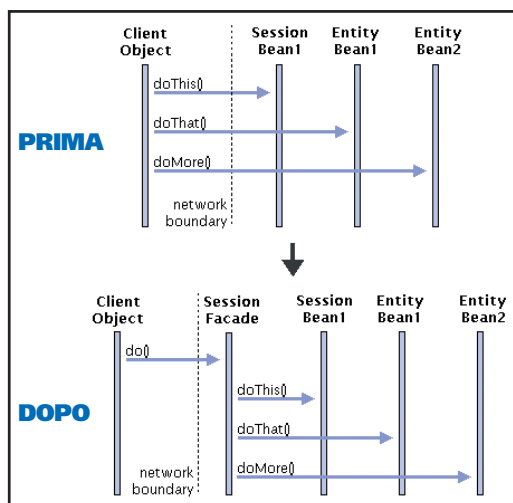



***Fig. 3:** Risultato dell'applicazione del pattern Session Façade*

L'implementazione di un *Session Façade* comporterà i seguenti vantaggi:

- si potranno creare client d'ogni tipo grazie al protocollo *RMI-IIOP* utilizzato dalla tecnologia EJB. Infatti, si potrà accedere al *Session Façade* tramite vari tipi di client, per esempio servlet, JSP, interfacce grafiche standalone, applicazioni da riga di comando, e persino client scritti in altri linguaggi mediante il supporto CORBA (protocollo *IIOP*);
- saranno ridotti gli oggetti visibili al client, semplificando l'interazione mediante un'interfaccia più semplice e chiara;
- risulterà facilitata la manutenibilità dell'applicazione nascondendo i dettagli degli oggetti di business. In pratica, la façade crea una sorta d'incapsulamento a livello di applicazione;
- il minor l'accoppiamento tra le classi del model

NOTA

I Java BluePrints definiscono un modello di programmazione applicativo per soluzioni *end-to-end* tramite l'utilizzo della piattaforma Java 2 Platform, Enterprise Edition (J2EE). Essi contengono linee guida, patterns, e codice per scenari reali della programmazione e della progettazione, che permettono la realizzazione di soluzioni robuste, scalabili, e portabili.

GLOSSARIO

GRANULARITÀ

Per metodo a granularità fine (*fine-grained*) intendiamo un metodo che punta ad ottenere risultati atomici, e poco significativi per una funzionalità di business. Un metodo a granularità grezza (*coarse-grained*), è invece un metodo che punta ad ottenere risultati "più efficaci".

e le altre classi si tradurrà in una migliore qualità del software;

- saranno ridotte le chiamate (via rete) a funzionalità a granularità fine. In questo modo la performance non saranno rallentate da frequenti invocazioni di metodi remoti.

Per quanto riguarda il nostro motore di ricerca, di seguito riportiamo il codice con cui l'oggetto *SearchEngineDelegate*, accede all'*EJB Session Façade* che abbiamo chiamato *SearchEngineEJB*. Ricordiamo che *SearchEngineDelegate* è un'implementazione del pattern *Business Delegate* ed era l'oggetto a cui la servlet delegava l'esecuzione dei suoi metodi di business. Come si può vedere dal codice seguente, anche il *Business Delegate* fa una delega, ma questa volta remota, al *Session Façade:*

```
public Collection standardSearch(String searchText) throws
  SearchEngineException, RemoteException
  {
    SearchEngineEJB searchEngineEJB =
                                        getSearchEngineEJB();
    try {
      return searchEngineEJB.standardSearch(
                                              searchText);
    }
    catch (RemoteException ex)
    {
      return new Vector();
    }
}
```

GLOSSARIO

PATTERN GOF

***Gof* sta per *Gang of Four.* Questo è il nomignolo che viene solitamente affibbiato ai quattro autori *(Gamma, Helm, Johnson, Vlissides)* del primo libro ufficiale che ha iniziato la categorizzazione dei pattern. Vengono detti *"pattern GoF"* i primi 23 pattern fondamentali descritti nel loro celeberrimo testo. Tra questi: *Singleton, Composite, State, Command, Proxy* etc...**

Evitiamo di riportare l'altro metodo di business (denominato *countrySearch()*) che permette di fare ricerche solo su siti di lingua italiana, avendo esso un'implementazione del tutto simile a *standardSearch()*.
A questo punto possiamo anche analizzare il codice d'implementazione del *Session Façade (SearchEngineEJB).*

```
public Collection standardSearch(String searchString)
                                                      throws
  SearchEngineException
  {
     DAOFactory df = DAOFactory.getDAOFactory(1);
     SearchEngineDAO dao = df.getSearchEngineDAO();
     return dao.standardSelect(searchString);
  }
```

Il codice è semplice: l'accesso alla base dati per la ricerca richiesta è eseguito da un oggetto che viene definito *dao*. Questo nome non è stato assegnato casualmente. Infatti, stiamo per introdurre un altro pattern molto importante: il pattern *Data Access Object (DAO).*

UN PATTERN PER L'ACCESSO AI DATI

Nella tecnologia J2EE, solitamente l'accesso alla base dati è gestito da *Entity Bean* (qualche volta con il supporto di un *Connector*). È anche possibile inserire direttamente codice JDBC all'interno di session bean, per accedere direttamente alla base dati. In alcuni casi però, conviene implementare il pattern *DAO* piuttosto che *Entity Bean*, o altre "soluzioni artigianali". Il pattern *Data Access Object*, infatti, assegna ad alcune classi (legate tra loro dall'ereditarietà, classi *DAO*), la responsabilità dell'accesso al meccanismo di immagazzinamento dati. In particolare, come per gli *Entity Bean*, è possibile creare una classe *DAO* per ogni tabella del database. Esistono varie strategie per implementare il pattern in questione. La prima strategia che descriveremo è chiamata *"Factory Method for Data Access Objects"*. È consigliata quando è noto l'*RDBMS* da utilizzare, e si è certi che non ci saranno migrazioni su altri meccanismi di storage.

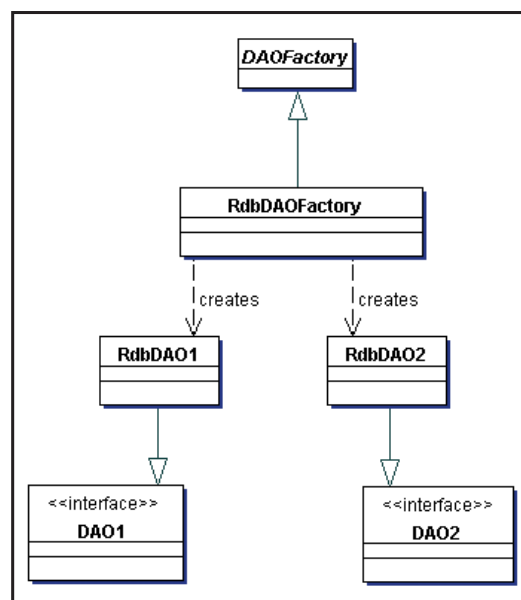



Fig. 4: **Class Diagram del modello del pattern DAO con strategia Factory Method**

Basandosi sul pattern *Factory Method*, si crea una classe che fa da *Factory* ai vari *DAO* dell'applicazione.
Per esempio, considerando che *SearchEngineDAO* è una superclasse astratta di *MySQLSiteDAO*, potremmo creare la seguente classe:

```
public class MySQLDAOFactory
{
  public SearchEngineDAO getSearchEngineDAO()
   {
     return new MySQLSiteDAO();
   }
}
```

ABSTRACT FACTORY FOR DAO

Anche se, nel caso del nostro motore di ricerca, questa strategia sarebbe stata più che sufficiente, ne utilizzeremo una più sofisticata, denominata *"Abstract Factory for Data Access Objects"*. Si tratta dell'applicazione del famoso pattern *GoF* noto come *Abstract Factory*, al problema dell'accesso ai dati.
La progettazione diventa un po' più complicata, ma i vantaggi sono evidenti. La sua implementazione rende l'applicazione il più possibile indipendente dal sistema di storage. In realtà, come è noto la tecnologia *JDBC* permette già di scrivere applicazioni indipendenti dalla base dati. Il costo da pagare è di non poter utilizzare i particolari dialetti delle varie distribuzioni, ma solo *ANSI SQL 2*. Per alcune applicazioni può rappresentare un vincolo troppo penalizzante.
Con il *Factory Method* è possibile creare diversi

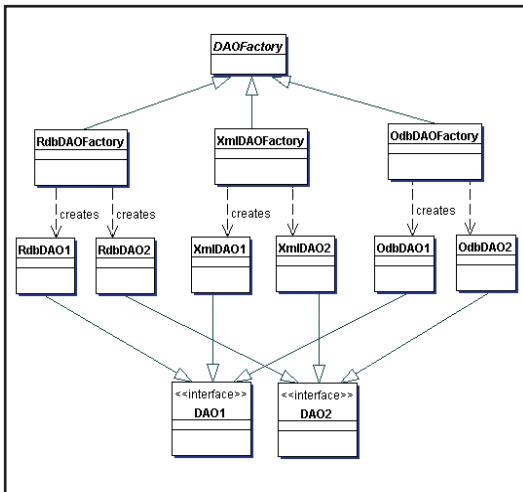



Fig. 5: ***Class Diagram del modello del pattern DAO con strategia Abstract Factory***

DAO che accedono a diverse tabelle, ma il database deve essere unico e stabilito. Anche se la maggior parte delle applicazioni *web-enterprise* si basano su database, è possibile che, con l'evolversi dei requisiti e delle tecnologie, le applicazioni possano sfruttare diversi meccanismi di storage, come repository *LDAP*, file XML, database object oriented, sistemi di integrazione esterni, B2B tramite web services etc. Con l'implementazione del pattern *DAO* con strategia *Abstract Factory*, questi problemi possono essere risolti.
Questa strategia comporta la creazione di una superclasse astratta *DAOFactory*, che dichiara un metodo *factory* astratto *getSearchEngineDAO()* per la creazione di un oggetto *DAO*, una costante simbolica *MYSQL*, ed un metodo statico factory per la creazione di uno specifico oggetto *DAOFactory*. Nel nostro esempio utilizzeremo solo il database *MySQL*, e quindi gestiremo solo una costante:

```
package com.cdsc.pub.j2eepatterns.dao;

public abstract class DAOFactory {
    private final static int MYSQL = 1;

    public static DAOFactory getDAOFactory(
                                    int whichFactory){
        switch (whichFactory) {
            case MYSQL:
                return new com.cdsc.pub.j2eepatterns
                                .dao.MySQLDAOFactory();
            default:
                assert false: "Unable to determinate a
                                          DAOFactory";
        }
    }

    public abstract SearchEngineDAO
                            getSearchEngineDAO();
}
```

Nulla ci vieterà, in futuro, di aggiungere una costante *ORACLE*, o *XML_FILE*, e i relativi controlli in *getDAOFactory()*.
Rivediamo allora la classe *MySQLDAOFactory*, che ora estende la classe astratta *DAOFactory*. La sottoclasse concreta *MySQLDAOFactory*, definisce il metodo *getSearchEngineDAO()*, restituendo il giusto oggetto *DAO*. Inoltre, espone un metodo per ottenere la **connessione**. Quest'ultimo passaggio, sarebbe stato implementato meglio in un oggetto *Service Locator* relativo allo strato business. Avendo già trattato il pattern *Service Locator* nel precedente articolo, abbiamo preferito evitarne l'implementazione:

```
package com.cdsc.pub.j2eepatterns.dao;
import java.sql.*;

public class MySQLDAOFactory
    extends DAOFactory {
    public static Connection createConnection() {
    Connection conn = null;
    try {
      Class.forName(context.getInitParameter(
                                    "jdbc.driver");
      conn = DriverManager.getConnection(
                              "jdbc.database.url ");
    }
    catch (Exception ex) {
        ...
    }
    return conn;
  }
```

Per compilare e deployare il progetto, si consiglia di utilizzare un IDE come *JBuilder X* o *Eclipse* (con plug-in *Lomboz*). Come application server, è possibile scegliere tra più alternative: *Websphere Server*, *Weblogic Server* o la coppia *Tomcat – JBoss…*

I pattern J2EE, vengono descritti illustrando diverse strategie di implementazione. Non esiste un modo univoco per implementare un pattern. Il pattern *DAO* per esempio, con le sue strategie può adattarsi a risolvere il problema dell'accesso ai dati in contesti diversi.

```
public SearchEngineDAO getSearchEngineDAO()
 {
   return new MySQLSiteDAO();
 }
}
```

È chiaro che è possibile introdurre facilmente nuovi oggetti *DAO*, o inserendo nuovi metodi del tipo *getXXXDAO()*, o con la modifica del metodo *getSearchEngineDAO()* con un parametro in input e un costrutto condizionale per restituire il giusto *DAO*. L'interfaccia *SearchEngineDAO*, definisce i metodi che tutti i *DAO* dovranno implementare. Nel nostro caso ci sono solo due metodi, ma potrebbero essercene altri quali *insert, update, delete* etc..

```
package com.cdsc.pub.j2eepatterns.dao;

import java.util.Collection;
import com.cdsc.pub.j2eepatterns.exceptions
                                  .SearchEngineException;

public interface SearchEngineDAO {
 Collection standardSelect(String searchString)
                         throws SearchEngineException;

 Collection countrySelect(String searchString)
                         throws SearchEngineException;
}
```

La sottoclasse *MySQLSiteDAO*, definisce i metodi astratti ereditati. Sfruttando la libreria *JDBC* (che sarà importata solo e solamente nella classe *DAO*), eseguirà le giuste query sulla base dati.

```
public Collection standardSelect(String searchString)
                         throws SearchEngineException {
  String query =
     "SELECT DISTINCT * FROM SITE S, META_
        KEYWORDS MK WHERE MK.KEYWORD = ' " +
          searchString + " ' AND MK.SITE_ID=S.ID";
  return select(query);
}

public Collection countrySelect(String searchString)
                         throws SearchEngineException {
  String query =
     "SELECT DISTINCT * FROM SITE S, META_
        KEYWORDS MK WHERE MK.KEYWORD = ' " +
        searchString + " ' AND S.LANGUAGE='it' AND
                              MK.SITE_ID=S.ID";
  return select(query);
}
private Collection select(String query) throws
                               SearchEngineException {
  Connection conn = null;
  PreparedStatement cmd = null;
  ResultSet rs = null;
  Vector sites = new Vector();
  try {
    conn = MySQLDAOFactory.createConnection();
    cmd = conn.prepareStatement(query);
    rs = cmd.executeQuery();
    while (rs.next()) {
      int id = rs.getInt("ID");
      String address = rs.getString("ADDRESS");
      String title = rs.getString("TITLE");
      String language = rs.getString("LANGUAGE");
      Site site = new Site(id, address, title, language);
      sites.addElement(site);
    }
  }
  catch (SQLException sqle) {
    throw new SearchEngineException(
                      "MySQLSiteDAO.select()", sqle);
  }
  finally {
      closeAll(conn, cmd, rs);
  }
  return sites;
}
```

I metodi *standardSearch()* e *countrySearch()*, formattano la query e ne delegano l'esecuzione al metodo privato *select()*. Notiamo che, una volta ottenuti i risultati della query, viene creato un oggetto di tipo **Site** per ogni record restituito. Infine, tutti gli oggetti *Site* creati vengono messi in un *Vector* e restituiti al chiamante.
Anche in questo frammento di codice abbiamo utilizzato un pattern: il pattern noto come *"Transfer Object"*. Questo pattern è noto anche con altri nomi: *"Value Object" (VO)* e *"Data Transfer Object" (DTO)* e sarà argomento del prossimo paragrafo. Prima, però, completiamo l'argomen to con *DAO*. In questa classe abbiamo utilizzato una terza strategia per il pattern *DAO*, chiamata per l'appunto *"Transfer Object Collection Strategy"*.
Ricapitolando, l'implementazione del pattern *DAO* comporterà i seguenti vantaggi:

- possibilità di migrare su differenti database potendo utilizzare i vari dialetti proprietari senza vincoli;
- possibilità di migrare su differenti meccanismi di immagazzinamento dati oltre ai classici database;
- centralizzazione di tutto il codice di accesso ai dati in un unico strato. Per esempio, se si accede a database, le librerie *JDBC* devono essere importate solo nelle classi *DAO*;
- riduce la complessità del codice nei client, a

NOTA

Spesso un pattern non è altro che il risultato della composizione di più pattern. Il *Factory method* è uno di quei pattern fondamentali, che costituisce la base di molti altri pattern. Oltre al *DAO*, il *Factory Method* può essere implementato per esempio in una strategia del pattern *Singleton*. Un esempio di pattern composito è l'*MVC*, che quantomeno "contiene" lo *Strategy* e l'*Observer*.

Non è raro per uno sviluppatore utilizzare pattern inconsapevolmente. Migliore è la soluzione che "inventiamo", più è alta la probabilità di stare utilizzando un pattern. Gli stessi componenti della *Gang Of Four*, decisero di scrivere la loro opera dopo essersi accorti di utilizzare le stesse tecniche di progettazione. La loro conclusione fu la definizione e la categorizzazione dei primi pattern fondamentali.

cui è trasparente il meccanismo di immagazzinamento dati;
- fornisce una interfaccia object oriented, ed incapsula lo schema del database;

Ovviamente, il costo da pagare è un maggiore complessità dal punto di vista della progettazione dei *DAO*, e un maggior impegno di codifica.

VALUE OBJECT

Il pattern *Value Object* è in assoluto uno dei pattern J2EE più utilizzati, a volte inconsapevolmente. Spesso, un *Value Object* corrisponde ad un oggetto di business (come nel nostro caso). La sua implementazione comporta la creazione di una classe (nel nostro caso la classe *Site*) che garantisce il trasporto di tutti i dati dallo strato *Business* allo strato WEB (e viceversa), per l'assolvimento di funzionalità di *business*. Questa classe deve essere presente come implementazione in entrambi gli strati e i suoi oggetti saranno passati per copia. Infatti, il costruttore della classe *Site* che abbiamo utilizzato nel metodo *select():*

```
public Site(int id, String address, String title,
                                    String language)
```

ci permette di incapsulare tutte le informazioni che saranno poi visualizzate dalla JSP di presentazione. In questo modo, eviteremo di incrementare il traffico di rete per ottenere dati diversi in più chiamate. Un'alternativa (nel nostro caso) sarebbe quella di utilizzare direttamente stringhe piuttosto che oggetti *Site* ma, ovviamente, andremmo a limitare la flessibilità della nostra applicazione. Nel caso del nostro motore di ricerca, lo strato Business ha creato un *Value Object* per restituire risultati allo strato Web. È spesso utile creare un *Value Object* nello strato *Web*, per poi passarlo allo strato *Business*. Si pensi ad esempio all'inserimento o all'aggiornamento di un certo dato nel database.
La documentazione ufficiale *(Sun BluePrints)*, asserisce che un *Value Object* è un oggetto che è possibile creare appositamente per risolvere il problema del trasporto dati. È addirittura consigliata come strategia di dichiarare *public* le sue variabili d'istanza, evitando così di creare i tipici metodi *set – get* dell'incapsulamento.
Questo perché un *Value Object* dovrebbe essere solo un oggetto con la responsabilità di trasportare dati, non di verifica dei dati stessi. In questo modo, però, si incoraggia lo sviluppatore a violare le principali regole dell'*Object Orientation* come l'astrazione e l'incapsulamento, in nome delle performance. A nostro parere, anche se riteniamo che una buona performance sia una caratteristica irrinunciabile del software, non siamo convinti che questo debba obbligatoriamente essere in contrasto con i canoni object oriented.
Creare oggetti di business che siano anche *Value Object*, come fatto con la classe *Site*, dovrebbe comunque essere un buon compromesso. Ecco perché preferiamo chiamare questo pattern *Value Object*, che in italiano potremmo tradurre come "oggetto di valore", piuttosto che *Transfer Object*, ovvero "oggetto di trasferimento".

CONCLUSIONI

Ricapitolando, in questo articolo abbiamo trattato tre nuovi design pattern, applicabili allo strato *Business*. Dopo aver puntualizzato che le regole di business devono essere implementate all'interno del *Model*, abbiamo descritto tre noti pattern J2EE: il *Session Façade*, il *DAO* e il *Value Object*. Il pattern *Session Façade*, ha fornito al *Model* (e quindi all'applicazione), un'interfaccia chiara e facilmente accessibile ai client.
Il pattern *DAO* invece, ha introdotto un nuovo strato di software che potremmo chiamare "strato di integrazione" *(integration tier)*, per la gestione dell'accesso ai dati. Con l'implementazione del pattern *DAO*, è possibile rendere l'applicazione indipendente dalla suo modo di immagazzinare dati. Con la strategia *Abstract Factory* si può gestire addirittura l'accesso a sistemi di storage eterogenei. Infine il pattern *Value Object*, rappresenta un modo naturale per lo scambio di applicazioni tra gli strati *Web* e *Business*. In questi due articoli abbiamo cercato di dare dei consigli utili per creare applicazioni basate su J2EE, nel modo più corretto possibile. Per realizzare il nostro scopo abbiamo utilizzato alcuni dei più famosi pattern J2EE, così come consigliato dai Sun BluePrints.
L'utilizzo dei design pattern è sempre consigliabile, e lo è ancora di più quando si sviluppa in un ambiente con tecnologie eterogenee. Non è semplice infatti essere padroni completamente di tanti argomenti come quelli relativi a J2EE. Ed è quindi complesso creare meccanismi che non impatteranno in qualche modo su di un requisito non funzionale dell'applicazione, come la performance, la manutenibilità, o l'estensibilità. Inoltre, i pattern sono centinaia, ed ognuno di essi può in qualche modo semplificare la progettazione delle nostre applicazioni.
Ma forse stiamo sfondando una porta già aperta…

Claudio De Sio Cesari

BIBLIOTECA

- **DESIGN PATTERN** ***Gamma, Helm, Johnson, Vlissides.*** **(Addison-Wesley)**
- **J2EE BluePrints – design patterns:** http://java.sun.com/blueprints/patterns/catalog.html
- **Il libro più venduto sui pattern J2EE:** http://java.sun.com/blueprints/corej2eepatterns/index.html
- **Il pattern MVC "vero":** www.claudiodesio.com/ooa&d/mvc.htm

L'AUTORE

Claudio De Sio, è un consulente free-lance che si occupa di tecnologia Java, analisi, progettazione e architetture object oriented. Laureato in matematica, dal 1999 collabora con Sun Educational Services (come docente e mentore), ed altri importanti aziende dell'IT. Sul suo sito internet www.claudiodesio.com**, ha pubblicato diverse risorse didattiche gratuite, relative agli ambiti dove è specializzato.**

Istruiamo un sistema esperto e usiamolo in Java

Intelligenza artificiale

In questo articolo impareremo qualcosa sui sistemi esperti, usando uno strumento pratico: JESS e creeremo un'applicazione Java che si interfaccia a un sistema di intelligenza artificiale

Al di là di ogni tipo di approfondimento, quello che faremo da qui in poi sarà tentare di istruire un software a reagire come se fosse dotato di un'intelligenza propria. Per i nostri scopi utilizzeremo un tool chiamato JESS. L'idea è molto semplice: scriveremo uno script che costituirà la base di conoscenza del sistema, definendo alcune regole di base. Sarà un po' come fornire le regole di comportamento ad un bambino. In seguito, daremo queste regole in pasto a JESS che le elaborerà, autonomamente, basandosi sui nostri insegnamenti.

COME INIZIARE

Presupponiamo che abbiate installato sul vostro sistema un JDK recente. Scaricare JESS dal sito http://herzberg.ca.sandia.gov/jess/index.html, oppure prelevarlo dal CD allegato a ioProgrammo. Scompattare il file che contiene JESS in una qualunque directory. Avviare un prompt di msdos e portarsi nella directory contenente i file di JESS. Digitare:

```
java –cp jass.jar jess.Main
```

Comparirà il prompt della shell JESS

```
JESS>
```

Che vi invita ad inserire qualcuna delle istruzioni che abbiamo citato in questo articolo. In alternativa potete provare uno dei numerosi esempi allegati al prodotto.
Il comando è:

```
jess –cp jess.jarr jess.Main examples/
                                 sticks.clb
```

il file sticks.clb contiene una serie di istruzioni identiche a quelle di cui parleremo in questa sede.

COSA È JESS

JESS *(Java Expert System Shell)* è una shell, ovvero un sistema a linea di comando, che permette di definire un sistema esperto. Creato da Ernest Friedman-Hill, JESS si è ispirato a CLIPS, un'altra shell per sistemi esperti. Scritto interamente in Java, JESS permette di avere a disposizione un motore inferenziale all'interno del proprio programma, che si appoggia ad una base di conoscenza (Knowledge Base). Questa shell mette a disposizione un completo ambiente per definire degli script con sintassi simile a quella del Lisp, grazie ai quali possiamo definire la base di conoscenza. Questa viene poi elaborata da Jess tramite un suo algoritmo, denominato "Rete", e quindi ci permette di inferire delle informazioni partendo da un problema iniziale. Per rappresentare la conoscenza in JESS abbiamo 3 diversi modi:

1. **Regole**
2. **Funzioni**
3. **Programmazione OO**

Una regola è qualcosa di molto simile al costrutto IF *THEN* dei linguaggi procedurali. In questi linguaggi *IF THEN* funziona nella seguente maniera:

```
IF è_vero_qualcosa
THEN fai_qualcosa
```

Ora, nella shell di JESS dovremo prima di tutto affermare quello che viene chiamato un fatto, ovvero una affermazione vera

```
Jess> (assert umano Socrate)
```

In questo modo abbiamo memorizzato il fatto che Socrate sia un essere umano. Ora definiamo la regola secondo la quale se Socrate è umano verrà stampato a schermo la stringa "Socrate è mortale"

```
Jess> (defrule mortaleSocrate
    (umano Socrate)
    =>
    (printout t "Socrate è mortale" crlf);
```

Abbiamo così definito la regola in JESS. Andando ad inserire il comando (*run*) nella shell verrà stampata esattamente la stringa che abbiamo inserito, perchè viene usato la regola che noi abbiamo definito. Questo succede perchè precedentemente, abbiamo

inserito il "fatto" che Socrate sia umano, quindi JESS, avendo a disposizione questa verità, applica la regola definita come *"mortaleSocrate"*, scrivendo a schermo con la funzione predefinita *"printout"* la stringa immessa nella definizione. Esistono altre funzioni predefinite come *printout* all'interno di JESS che possiamo utilizzare. In realtà, tutto quello che scriviamo in JESS sembra una chiamata a funzione. Ad esempio, quando vogliamo che la nostra shell interpreti una semplice addizione come *1+7*, dobbiamo formattare per ricevere immediatamente la risposta senza aver definito alcuna funzione

```
Jess> (+ 1 7)
8
```

Una lunga lista di funzioni può essere trovata nella documentazione ufficiale di JESS, grazie alla quale possiamo sapere cosa è già implementato senza dover riscrivere del codice. Possiamo comunque definire le nostre funzioni tramite il costrutto deffunction. Una volta definita, potrà essere richiamata all'interno della shell. Vediamo, ad esempio, come definire una funzione per calcolare l'area di un triangolo

```
Jess> (deffunction area_triangolo (?a ?b)(/ (* ?a ?b) 2))
TRUE
Jess> (area_triangolo 5 7)
17.5
```

La sintassi è abbastanza intuitiva, nella dichiarazione abbiamo gli argomenti passati alla funzione *?a* e *?b* e dopo elaboriamo i dati restituendo l'area del triangolo. Grazie alla funzione *batch*, possiamo richiamare dei file di testo in cui definiamo le nostre funzioni senza doverli riscrivere ogni volta. Oltre alle feature già spiegate, JESS permette di definire dei veri e propri oggetti da richiamare e usare all'interno dei propri programmi. Tramite il costrutto *deftemplate*, diamo la definizione dell'oggetto, del quale poi setteremo i campi all'interno del nostro programma

```
Jess> (deftemplate cellulare
  "Un cellulare."
  (slot marca)
  (slot modello)
  (slot anno (type INTEGER))
  (slot color (default bianco)))
TRUE
Jess> (assert (cellulare (marca Nokia) (modello 7600)
                                (anno 2004) (color nero)))
<Fact-0>
Jess> (facts)
f-0   (MAIN::cellulare (marca Nokia) (modello 7600)
```

NOTA

Nell'intelligenza artificiale possiamo distinguere due diverse correnti di pensiero: top-down e bottom-up. Il primo è quello riguardante l'intelligenza artificiale classica, con la definizione di strutture logiche ad hoc per un determinato problema. Bottom-up invece contraddistingue le metodologie che cercano di creare dell'intelligenza semplicemente permettendo all'agente di venire a contatto con un ambiente esterno a lui sconosciuto (ad esempio le Reti Neurali).

SISTEMI ESPERTI E INTELLIGENZA ARTIFICIALE

L'intelligenza artificiale è quella disciplina che tenta di costruire delle entità intelligenti basandosi su determinati algoritmi. Una delle prime definizioni di Intelligenza Artificiale è quella di Marvin Minsky secondo il quale "È intelligenza artificiale quel settore dell'informatica che cerca di riprodurre nei computer quel tipo di comportamenti che, quando sono assunti dagli esseri umani, vengono generalmente considerati frutto della loro intelligenza". Questa disciplina è composta da molteplici settori, ognuno dei quali cura una particolare caratteristica della rappresentazione della conoscenza. Il campo dei Sistemi Esperti è una delle varie applicazioni dell'intelligenza artificiale. Per *Sistema Esperto* si intende un programma di intelligenza artificiale che raggiunge una certa competenza in un settore, avendo quindi le stesse potenzialità di un essere umano nella risoluzione dei problemi per quel determinato settore. L'interesse dietro ai Sistemi Esperti è chiaramente giustificato: si possono definire dei programmi che, partendo da informazioni di base, riescono a dare un risultato a problemi complessi, avendo quindi un sistema automatizzato che li risolve. I campi di applicazione dei Sistemi Esperti possono essere tanti quanti i problemi che possono essere definiti in un programma, quindi infiniti. Per costruire un Sistema Esperto c'è la necessità che lo sviluppatore stringa una stretta collaborazione con l'utente del sistema, identificando i problemi e gli obiettivi da realizzare. Poi sarà compito dello sviluppatore formalizzare questi dati, definendo quindi la struttura del Sistema Esperto. La persona che svolge questo compito viene di solito indicata come *Ingegnere della Conoscenza,* colui che partendo dal sistema definisce il Sistema Esperto da implementare. A questo punto entrano in gioco i linguaggi di AI, con i quali lo sviluppatore deve implementare il vero e proprio programma.

Settori dell'intelligenza artificiale.

Servono quindi degli strumenti che possano permette la definizione di problemi e che permettano di inferire un risultato basandosi sulle condizioni iniziali di problema e sull'obiettivo che si vuole raggiungere. Le tipologie di linguaggi di programmazione che vengono utilizzati nell'Intelligenza Artificiale sono principalmente due: funzionali e logici. Nei linguaggi funzionali il programma, è definito da una serie di definizioni di funzioni e l'esecuzione si riduce al calcolo del valore di un'espressione. Il LISP (LISt Processing) è un linguaggio funzionale che viene spesso utilizzato nell'Intelligenza Artificiale. Ci sono poi i linguaggi logici, nei quali il programma è formato da una serie di affermazioni. Quando poi viene avviato il programma questi linguaggi controllano se una certa formula è conseguenza logica delle informazioni a sua disposizione. Il PROLOG (PROgramming in LOGic) è un linguaggio logico a disposizione degli sviluppatori di AI. Esistono diverse implementazioni di questi linguaggi e di shell per costruire un sistema esperto.

NOTA

Un esempio di Sistema Esperto è MOMA, realizzato da Enea per monitorare l'adeguamento alle normative europee delle strutture fognarie e depurative.
http://wwwamb.bologna.enea.it/moma/

JESS, dalla versione 5.0, ha il supporto sia alla forward chaining che alla backward chaining. Forward chaining (concatenamento in avanti) è praticamente il meccanismo che, partendo da fatti noti e applicando le regole definite, giunge a nuovi fatti. Il backward chaining (concatenamento all'indietro) è invece il meccanismo che, partendo da un possibile fatto cerca le regole che lo possano dedurre.
http://herzberg.ca.sandia.gov/jess

```
                                    (anno 2004) (color nero))
For a total of 1 facts.
```

Altre interessanti feature di Jess sono quelle di poter importare classi Java in codice Jess e di poter richiamare una shell Jess all'interno di programmi Java.

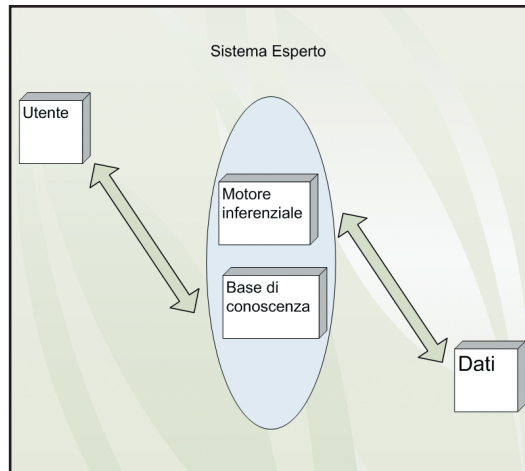



Fig. 2: ***Struttura di un Sistema Esperto***

Tutto ciò permette da una parte di avere un linguaggio di scripting come Jess molto potenziato dalle classi Java, dall'altra un programma Java che può avere a disposizione un motore di inferenza semplice ed efficiente.

IL PROBLEMA DELLA CAPRA, DEL CAVOLO E DEL LUPO

Vediamo ora di risolvere uno dei problemi classici dell'intelligenza artificiale, ovvero quella capra, del cavolo e del lupo. Un contadino deve portare da una riva all'altra di un fiume una capra, un cavolo e un lupo con una barca che può trasportare solo due cose alla volta. Quindi il contadino può trasportare soltanto un'altra cosa oltre a se stesso. C'è però il problema che la capra non può essere lasciata sola con il cavolo altrimenti lo mangia e, per lo stesso motivo, il lupo non può essere lasciato con la capra. Prima di tutto, dobbiamo definire lo stato in cui ci possiamo trovare nel seguente modo

```
(deftemplate MAIN::status
   (slot profondita_ricerca)
   (slot precedente)
   (slot posizione_contadino)
   (slot posizione_lupo)
   (slot posizione_capra)
   (slot posizione_cavolo)
   (slot ultima_mossa)
)
```

In questo modo abbiamo definito un oggetto di fondamentale importanza per monitorare lo status durante l'esecuzione del nostro programma.
Dobbiamo poi dare al nostro programma uno stato iniziale dal quale partire per risolvere il problema definendo dei fatti

```
(deffacts MAIN::posizioni_iniziali
  (status (profondita_ricerca 1)
          (precedente nessun_precedente)
          (posizione_contadino riva-1)
          (posizione_lupo riva-1)
          (posizione_capra riva-1)
          (posizione_cavolo riva-1)
          (ultima_mossa nessuna_mossa)))
(deffacts MAIN::opposti
  (opposto_della riva-1 riva2-2)
  (opposto_della riva-2 riva-1))
```

Ora il nostro programma conosce le posizioni iniziali dei personaggi, ovvero tutti sulla prima riva, e il concetto di "opposto", avendo definito rispettivamente il concetto di opposto per la riva 1 e la riva 2.
È arrivato quindi il momento di definire le mosse che il nostro programma potrà fare durante l'esecuzione per arrivare alla soluzione. Vediamo ad esempio la definizione della mossa del contadino che trasporta sulla barca il lupo

```
(defrule MAIN::mossa_col_lupo
  ?node <- (status (profondita_ricerca ?num)
  (posizione_contadino ?fs)
  (posizione_lupo ?fs)
  )
  (opposite-of ?fs ?ns)
  =>
  (duplicate ?node (profondita_ricerca (+ 1 ?num))
  (precedente ?node)
  (posizione_contadino ?ns)
  (posizione_lupo ?ns)
  (ultima_mossa fox))
  )
```

Questa regola controlla lo status e lo aggiorna per quanto riguarda le posizioni e soprattutto salvandosi lo stato precedente, di modo che, se il nostro programma nell'elaborazione del risultato incappasse in una serie passaggi sbagliati, potremmo tornare indietro e provare un'altra strada per raggiungere l'obiettivo. Chiaramente il programma ancora non sa quali sono i vincoli che non devono essere violati. Noi, analizzando il problema, sappiamo che non possiamo lasciare da soli lupo e capra altrimenti quest'ultima viene mangiata.
Per questo motivo dobbiamo definire delle regole come la seguente:

```
(defrule CONSTRAINTS::lupo_mangia_capra
```

```
                    (declare (auto-focus TRUE))
 ?node <- (status (posizione_contadino ?s1)
            (posizione_lupo ?s2&~?s1)
            (posizione_capra ?s2))
 =>
 (retract ?node))
```

Con questa regola noi cancelliamo, grazie alla funzione retract, un nodo nella nostra elaborazione se in questo nodo si verifica che il lupo sia rimasto da solo con la capra senza il contadino. In questo modo JESS cancella dal nostro albero di tentativi un nodo che non ci porta alla soluzione.
Ora dobbiamo schematizzare quale sia la soluzione e come stampare a schermo tutti i passi che devono essere fatti. La soluzione viene definita come una regola standard nella quale ci rendiamo conto che lo status è quello dichiarato dall'obiettivo del problema, ovvero tutti i protagonisti si trovano sulla riva 2.

```
(deftemplate SOLUTION::mosse
  (slot id)
  (multislot lista-mosse))

(defrule SOLUTION::riconosci_soluzione
 (declare (auto-focus TRUE))
 ?node <- (status (precedente ?parent)
            (posizione_contadino riva-2)
            (posizione_lupo riva-2)
            (posizione_capra riva-2)
            (posizione_cavolo riva-2)
            (ultima_mossa ?move))
 =>
 (retract ?node)
 (assert (mosse (id ?parent) (lista-mosse ?move))))
```

Ora quello che resta da fare è far stampare a schermo le mosse che vengono eseguite da JESS per raggiungere l'obiettivo che abbiamo inserito. Questo lo possiamo fare scrivendo una regola che semplicemente, scorrendo la lista di mosse presenti nella base di conoscenza, le stampi a schermo, specificando di volta in volta che significato ha quella mossa

```
(defrule SOLUTION::stampa_soluzione ?mv <-
     (moves (id no-parent) (moves-list no-move $?m))
 =>
 (retract ?mv)
 (printout t crlf "Soluzione trovata: " crlf crlf)
 (bind ?length (length$ ?m))
 (bind ?i 1)
 (bind ?shore riva-2)
 (while (<= ?i ?length)
   (bind ?thing (nth$ ?i ?m))
   (if (eq ?thing alone)
     then (printout t "Il contadino si muove da solo
                         verso la " ?shore "." crlf)
     else (printout t "Il contadino si muove con "
                 ?thing " verso la " ?shore "." crlf))
   (if (eq ?shore riva-1)
     then (bind ?shore riva-2)
     else (bind ?shore riva-1))
   (bind ?i (+ 1 ?i))))
```

Ora non ci resta altro che inserire le funzioni (*reset*) e (*run*) e abbiamo il programma da mandare in pasto alla nostra shell per poter vedere la soluzione. Quando scarichiamo JESS, troviamo un esempio uguale a questo nella cartella examples con il nome *dilemma.clp*. Per avviare da riga di comando questo programma, non dobbiamo far altro che richiamare la shell di JESS e ci verrà stampato a schermo il risultato dell'elaborazione.

```
[federico@2828 Jess61p7]$ java -classpath jess.jar
                        jess.Main examples/dilemma.clp
Jess, the Java Expert System Shell
Copyright (C) 2001 E.J. Friedman Hill and the Sandia
                                          Corporation
Jess Version 6.1p7 5/7/2004
This copy of Jess will expire in 26 day(s).
Solution found:
Farmer moves with goat to shore-2.
Farmer moves alone to shore-1.
Farmer moves with fox to shore-2.
Farmer moves with goat to shore-1.
Farmer moves with cabbage to shore-2.
Farmer moves alone to shore-1.
Farmer moves with goat to shore-2.
```

RICHIAMARE JESS IN PROGRAMMI JAVA

Volendo usare JESS all'interno di applicazioni Java dobbiamo prima di tutto conoscere alcune classi fondamentali per interagire con questa shell come *jess.Value*, *jess.Context* e *jess.Rete*. La classe *jess.Value* è quella che rappresenta il valore che ci viene restituito alla fine dell'elaborazione effettuata da

```
federico@2828 Jess61p7]$ java -classpath jess.jar jess.Main esempio.clp

Jess, the Java Expert System Shell
Copyright (C) 2001 E.J. Friedman Hill and the Sandia Corporation
Jess Version 6.1p7 5/7/2004

This copy of Jess will expire in 25 day(s).

Soluzione trovata:

Il contadino si muove con capra verso la riva-2.
Il contadino si muove da solo verso la riva-1.
Il contadino si muove con lupo verso la riva-2.
Il contadino si muove con capra verso la riva-1.
Il contadino si muove con cavolo verso la riva-2.
Il contadino si muove da solo verso la riva-1.
Il contadino si muove con capra verso la riva-2.
```

Fig. 2: **Output del programma caricato da shell**

CLIPS (C Language Integrated Production System) è un tool per creare sistemi esperti, sviluppato nel 1985. È stato utilizzato per sviluppare molti prodotti e attualmente è disponibile per il download la versione 6.2
http://www.ghg.net/clips/CLIPS.html

La JSR 94 (javax.rules API) ha introdotto in Java la programmazione basata sulle regole. Una delle implementazioni aderenti allo standard di questa JSR è appunto JESS.
http://jcp.org/aboutJava/communityprocess/review/jsr094/
www.theserverside.com/articles/article.tss?l=Jess

L'AUTORE

Federico Paparoni è laureato in Ingegneria Informatica. È specializzato nello sviluppo di applicazioni client-server per terminali mobili e lavora come progettista software per una Mobile Company che offre la sua consulenza ai gestori di telefonia e alle major del settore ICT. I suoi interessi principali riguardano le piattaforme J2ME e J2EE. È Admin di www.JavaStaff.com, portale per la divulgazione delle tecnologie Java, e può essere contattato all'email federico.paparoni@javastaff.com

Jess. *jess.Context* rappresenta invece il contesto durante l'esecuzione di una shell, incorporando in sé informazioni riguardanti funzioni e variabili presenti nell'environment. Infine abbiamo *jess.Rete*, classe che rappresenta il vero e proprio motore inferenziale. Istanziando questa classe inizia l'interazione con la shell come possiamo vedere nell'esempio seguente

```
import jess.*;
 public class AreaTriangolo
 {
   public static void main(String a[])
   {
     try
       {
         //Istanza della shell
         Rete r = new Rete();
         //comunichiamo alla shell la definizione di una
                                                funzione
         r.executeCommand("(deffunction area_
                     triangolo (?a ?b) (/ (* ?a ?b) 2) )");
         //richiamiamo la funzione appena definita e
                     salviamo il valore in un oggetto Value
         Value v = r.executeCommand("(
                                    area_triangolo 3 4)");
         // recuperiamo ora questo valore specificando che
         // è un valore intero e lo stampiamo a schermo
         System.out.println(v.intValue(r.getGlobalContext(
                                                    )));
     }
     catch (Exception e)
     {
         System.err.println(e);
     }
   }
}
```

```
federico@2828 Jess61p7]$ java -classpath jess.jar jess.Main swing.clp

Jess, the Java Expert System Shell
Copyright (C) 2001 E.J. Friedman Hill and the Sandia Corporation
Jess Version 6.1p7 5/7/2004

This copy of Jess will expire in 26 day(s).
Hello, World!
```

Hello

Fig. 3: ***Esempio di Hello World importando le librerie SWING***

Per poter importare le classi di Jess dobbiamo chiaramente aver scaricato il JAR dal sito ufficiale e averlo inserito all'interno del nostro classpath.

IMPORTARE CLASSI JAVA IN JESS

Un'altra importante caratteristica di JESS è il fatto di potersi interfacciare con delle classi Java, mentre scriviamo degli script. Praticamente abbiamo accesso alle classi Java standard e inoltre ad altre classi che possiamo definire noi. Queste verranno poi richiamate negli script come classi esterne e utilizzate normalmente. Vediamo un semplice esempio. Noi abbiamo la seguente classe Java

```
public class JessSimpleObject
{
   public JessSimpleObject() { }
   public int fattoriale(int i)
   {
      if (i==1)
          return i;
      else
          i=i*fattoriale(i-1);
      return i;
   }
}
```

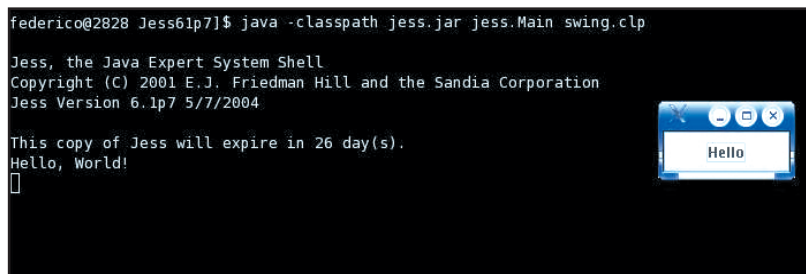
Per richiamare questa classe all'interno del nostro programma JESS dobbiamo creare un file di testo con le seguenti istruzioni

```
(defglobal ?*external-class* = 0)
(deffunction initialize ()
(bind ?*external-class* (new JessSimpleObject)))
(initialize)
(printout t "Il fattoriale di 4 è: " (?*external-class*
                                      fattoriale 4) crlf)
```

Richiamando questo file con il metodo (batch nomefile) riceviamo in output la stringa "Il fattoriale di 4 è: 24".
Potendo quindi richiamare classi Java standard possiamo anche creare delle semplici GUI con le API grafiche AWT, sviluppando così dei semplici programmi che sfruttino la parte grafica di Java.

CONCLUSIONI

Quello che abbiamo visto in questo articolo è un semplice esempio per l'utilizzo di JESS.
D'altronde è facile ricondurre molti problemi in termini di regole da rispettare e obiettivi da raggiungere, descrivibili in maniera semplice come abbiamo fatto qui. In tal caso JESS può essere un'ottima soluzione, anche per includerlo nei nostri programmi Java, avendo così semplicemente a disposizione un motore inferenziale, che ci permette di risparmiare molte linee di codice.
Infatti un tool come JESS può essere utile per quanto riguarda la sperimentazione e soprattutto per suddividere la logica di business dal programma.

Federico Paparoni

Un'introduzione al nuovo IDE open-source

Sviluppare in Java con Eclipse 3

IV parte

Vi siete mai chiesti come fanno a rilasciare le applicazioni in tutte le lingue? Vi mostreremo come realizzare una interfaccia che si adatta al paese in cui viene visualizzata... senza doverla riprogrammare!

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
Programmazione Java

Software
Eclipse 3.0.1 e JDK 1.4.2 o superiore su qualunque sistema operativo supportato (Linux, HP-UX, Solaris, AIX, Windows, MacOS)

Impegno

Tempo di realizzazione

Le applicazioni devono spesso far fronte al problema dell'internazionalizzazione e della localizzazione, per divenire abbastanza flessibili da poter essere facilmente e velocemente adattate a lingue diverse da quella originale. Oggigiorno in modo particolare questo tipo di problema è molto sentito e Java, linguaggio relativamente giovane, è già nato con una certa adattabilità linguistica in mente. Offre infatti diversi strumenti che permettono al programmatore accorto di generare output che verrà automaticamente localizzato dalla macchina virtuale e supporta nativamente il set di caratteri UNICODE, con cui si possono visualizzare praticamente i simboli di tutte le principali lingue conosciute.

LOCALIZZARE UN'APPLICAZIONE

L'**internazionalizzazione** di una applicazione è la (ri)scrittura della stessa in modo che sia possibile renderla disponibile per diverse parti del mondo in lingua locale senza necessità di modifica del software, ma con la semplice aggiunta di materiale tradotto in lingua (testi ed immagini, principalmente). Questa aggiunta è detta invece **localizzazione**: con questo termine si intende infatti il processo che rende un'applicazione internazionalizzata disponibile in una nuova lingua. Solitamente si tende a pensare che tali termini implichino molta traduzione di testo e un po' di lavoro di codice, ma non bisogna dimenticare alcune considerazioni di grande importanza: da un lato è vero che grossa parte dello sforzo deve essere dedicata a tradurre gli elementi testuali del prodotto software, ma pensate anche ad eventuali immagini che contengono testo, ad eventuali suoni in lingua, ed alle stringhe parametrizzate, che devono essere studiate con cura affinché sia possibile includere i vari parametri nelle traduzioni tenendo conto di singolari, plurali, eventuali duali, termini maschili, femminili e neutri, coniugazioni verbali ed inflessioni di aggettivi e sostantivi. Inoltre, un lavoro ben fatto non può prescindere dalle differenze nazionali relative alla visualizzazione di numeri, date, orari e valute. Al di là dell'ovvio, per esempio, sapevate che in giapponese i numeri non si raggruppano ogni 3, bensì ogni 4 cifre? Un'applicazione che voglia essere internazionalizzata dovrà quindi essere pronta a scrivere diecimila come 1.0000 e non 10.000 quando la lingua scelta è il giapponese. Vedremo quindi quali strumenti ci offre Java per supportare l'internazionalizzazione nei nostri programmi, per poi passare a vedere come Eclipse ci aiuta ad utilizzare tali strumenti.

GLI STRUMENTI OFFERTI DA JAVA

La cosa più importante per poter offrire supporto all'internazionalizzazione e localizzazione di un prodotto informatico è sicuramente la coscienza di quale lingua interessi all'utente del nostro software. Ed a questo proposito Java utilizza il concetto di "*locale*" (il termine è inglese, la "e" non si pronuncia), con cui si intende un assetto linguistico e nazionale specifico (per esempio, l'italiano dell'Italia rispetto a quello della Svizzera, o l'inglese britannico rispetto a quello americano). Una macchina virtuale Java gira sempre con una determinata *locale* che è detta la *locale* di sistema, e tutte le classi che si occupano di manipolare testo producono il loro risultato in base alla *locale* di sistema o ad una stabilita dal programmatore. Detto questo, la traduzione dei testi, seppur lungi dall'essere l'unico problema, è – come abbiamo accennato in precedenza – per lo meno uno dei principali. Java offre un pratico

sistema per recuperare testi da file di proprietà esterni al codice che vengono automaticamente selezionati in base alla *locale* prescelta.
Facciamo un semplice esempio.
Questo programma stampa tre messaggi all'avvio.

```
public class WithoutI18nSupport {
  static public void main(String[] args) {
    System.out.println("Hello. Welcome to our I18N
                                    application!");
    System.out.println("How are you? Have you ever
                                heard of I18N before?");
    System.out.println("Goodbye. See you soon...");
  }
}
```

```
Console
<terminated> WithoutI18nSupport [Java Application] C:\Programmi\Java\j2sdk1.4.2_03\bin\j
Hello. Welcome to our I18N application!
How are you? Have you ever heard of I18N before?
Goodbye. See you soon...
```

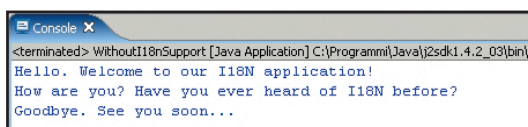



Fig. 1: *Il risultato prodotto da WithoutI18nSupport*

Il risultato dell'esecuzione di questa banale applicazione la vedete in **Figura 1**. Ma adesso facciamola diventare internazionalizzata. Java ci consiglia di usare dei file di proprietà (detti *bundle* o *resource bundle*) in cui teniamo i testi delle stringhe da localizzare. Un file sarà il default (hanno tutti l'estensione *.properties*), mentre gli altri avranno attaccato al nome un suffisso relativo alla *locale* in cui sono tradotti (*_it_IT, _it_CH, _en_UK, en_US,* etc). Ad ogni messaggio necessario per l'applicazione sarà associata la stessa chiave che viene utilizzata per recuperarlo nei diversi file di proprietà. Così, per il nostro esempio, il file di default si chiamerà *i18n.properties* (il nome è arbitrario) e conterrà questi valori:

greeting Hello. Welcome to our I18N application!
farewell Goodbye. See you soon...
inquiry How are you? Have you ever heard of I18N before?

Poi possiamo creare il file per l'italiano, *i18n_it_IT.properties:*

greeting Ciao. Benvenuto nella nostra applicazione I18N!
farewell Ciao. A presto...
inquiry Come va? Hai mai sentito parlare di I18N prima?

E per tutte le altre lingue che vogliamo, per esempio il portoghese *(i18n_pt_BR.properties)*

greeting Oi. Bem vindo no programa de I18N da gente!
farewell Tschau. Até a próxima...
inquiry Como vai? Você já ouviu falar de I18N?

ed il giapponese (*i18n_jp_JA.properties*):

greeting こんにちわ、I18N アプリケーションにようこそ
farewell さようなら、またお会いしましょう
inquiry ごきげんいかがですか? 皆さんは I18N をご存知ですか?

A questo punto, non ci resta che usare la classe *ResourceBundle* del package *java.util* per recuperare le informazioni dai file invece di usare le stringhe dentro il codice. Per creare un *ResourceBundle* diamo il nome del file che ci interessa senza suffisso ed estensione (nel nostro caso *i18n*) ed il sistema provvederà a prendere la traduzione giusta ricercando il bundle con l'estensione che corrisponde alla *locale* di sistema. Il metodo *getString* della classe, poi, ci restituisce il messaggio a fronte della chiave desiderata. Ecco come diventa la nostra applicazione dopo l'internazionalizzazione.

```
public class WithI18nSupport {
  static public void main(String[] args) {
    ResourceBundle messages =
                    ResourceBundle.getBundle("i18n");
    System.out.println(messages.getString("greeting"));
    System.out.println(messages.getString("inquiry"));
    System.out.println(messages.getString("farewell"));
  }
}
```

Adesso l'applicazione è pronta per girare in una delle quattro lingue predisposte, ed aggiungerne una nuova (cioè, un'ulteriore localizzazione) comporta semplicemente l'aggiunta di un altro file di proprietà tradotto. Se eseguiste l'applica-

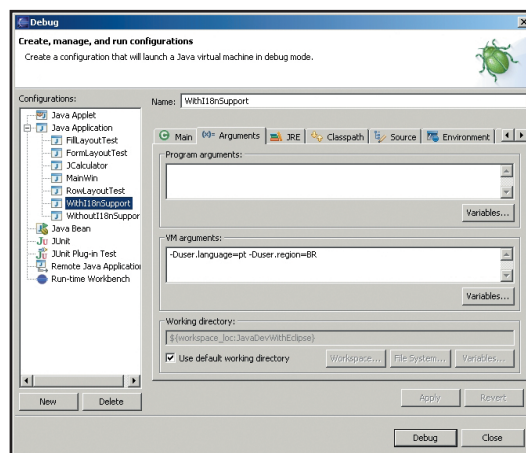



Fig. 2: *Come impostare la locale di sistema in Eclipse*

I18N E L10N

Per chi non abbia ancora molta dimestichezza con le problematiche legate all'internazionalizzazione (*I18N*) e alla localizzazione (*L10N*) possono risultare molto utili i link dedicati all'argomento dal famoso Java Tutorial
http://java.sun.com/docs/books/tutorial/i18n/
dal sito di Java
http://java.sun.com/j2se/corejava/intl/index.jsp

JVM E LOCALE

La macchina virtuale Java è avviata utilizzando la locale di default del sistema. È comunque possibile scegliere una locale diversa sotto cui fare girare le proprie applicazioni con l'opzione di comando

```
Duser.language=<lingua>
    -Duser.region=<paese>
```

oppure direttamente con il codice utilizzando il metodo statico *setDefault* della classe *Locale*, ad esempio

```
Locale.setDefault(
   new Locale("it", "IT"));
```

zione adesso probabilmente la vedreste in italiano se il vostro PC è in italiano, ma è comunque possibile impostare manualmente la *locale* di sistema della macchina virtuale Java con due definizioni specifiche (*user.language* e *user.region*) che si passano alla JVM con l'opzione *–D*. Se usate Eclipse, potete creare queste definizioni con l'apposita finestra di cui abbiamo già parlato in articoli passati e che si apre dal menu *Run->Run…* o *Run->Debug…* nella pagina *Arguments* (come in **Figura 2**, con l'effetto ottenuto visualizzato in **Figura 3**).

```
Console
<terminated> WithI18nSupport [Java Application] C:\Programmi\Java\j2sdk1.4.2_03\bin\javaw.exe (Nov
Oi. Bem vindo no programa de I18N da gente!
Como vai? Você já ouviu falar de I18N?
Tschau. Até a próxima...
```

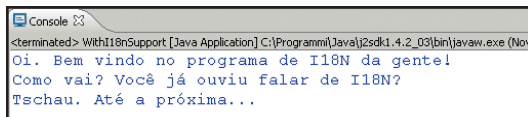



Fig. 3: ***Ora siamo in Brasile***

GLOSSARIO

L'ELENCO DELLE LOCALE

Per avere un elenco di tutte le locale disponibili sul vostro sistema, Java vi mette a disposizione il metodo statico *getAvailableLocales* sulle varie classi che supportano la formattazione localizzata. Esso vi restituisce un array di *Locale*. Così, per esempio, se volete sapere quali locale avete a disposizione per la formattazione delle date, potreste utilizzare codice del tipo *Locale list[] = DateFormat.getAvailableLocales();*

PROBLEMI DI CODICE: DATE E VALUTE

Prima di vedere come Eclipse ci aiuta con l'internazionalizzazione, vorrei accennare rapidamente ad alcune delle tecnologie che rendono Java particolarmente adatto a scrivere applicazioni multi-lingua e multi-cultura. In primo luogo, restando sul concetto dei bundle, dobbiamo aggiungere che con essi si possono anche creare degli insiemi di oggetti che non siano solo testo. Ciò viene incontro all'esigenza di internazionalizzare anche immagini, audio clip e dati di vario genere. È infatti possibile creare delle classi che estendano *ListResourceBundle* che vengono utilizzate con il metodo *getBundle* di *ResourceBundle* e che quindi vengono instanziate in base al loro suffisso linguistico (per approfondire vedi *http://java.sun.com/docs/books/tutorial/i18n/resbundle/list.html*). Oltre a testo, immagini e audio, però, avevamo già anche accennato ai problemi legati a numeri, date, valute e ore. Per superare le differenze su queste tipologie di dato, non è necessario memorizzare diverse volte lo stesso valore, ci basta formattare la stessa informazione in modo diverso a seconda della *locale* dell'utente. Insomma, il 23 di dicembre è sempre il 23 di dicembre, anche se viene scritto 12/23 negli Stati Uniti, 23/12 in Inghilterra e 23.12 in Germania. Così Java ci offre una serie di classi che si occupano proprio di formattare determinate tipologie di

Stile	Data (en_US)	Ora (en_US)
DateFormat.DEFAULT	10-Apr-98	3:58:45 PM
DateFormat.SHORT	4/10/98	3:58 PM
DateFormat.MEDIUM	10-Apr-98	3:58:45 PM
DateFormat.LONG	April 10, 1998	3:58:45 PM PDT
DateFormat.FULL	Friday, April 10, 1998	3:58:45 o'clock PM PDT

Tabella 1: I parametri per definire il formato di data e ora

```
Console
<terminated> I18nNumbers [J
$9,876,543.21

Console
<terminated> I18nNumbers [Java
R$ 9.876.543,21

Console
<terminated> I18nNumbers [Ja
€ 9.876.543,21
```

Fig. 4: ***A confronto la rappresentazione di una somma di denaro in vari paesi***

dato a seconda della *locale* desiderata. Cominciando dai numeri, abbiamo a disposizione *NumberFormat*, con una serie di metodi statici che restituiscono istanze per la formattazione di percentuali (*getPercentInstance*), numeri interi (*getIntegerInstance*), valute monetarie (*getCurrencyInstance*) e numeri generici (*getNumberInstance*) in base alla *locale* di sistema o ad una data come parametro. Ecco un esempio di come formattare un numero in base alla *locale* di sistema (le **Figure 4a**, **b** e **c** vi mostrano lo stesso codice eseguito sotto le *locale en_US, pt_BR* e *it_IT*):

```
public class I18nNumbers {
  public static void main(String[] args) {
    Double currency = new Double(9876543.21);
    NumberFormat currencyFormatter;
    String currencyOut;
    currencyFormatter =
                NumberFormat.getCurrencyInstance();
    currencyOut = currencyFormatter.format(currency);
    System.out.println(currencyOut); }
}
```

Passiamo invece alle date e agli orari. Il principio è lo stesso: abbiamo una classe *DateFormat* con tre metodi statici che formattano date (*getDateInstance*), orari (*getTimeInstance*) o entrambi (*getDateTimeInstance*). Oltre al parametro opzionale che riguarda la *locale* da utilizzare (per default è quella di sistema), questi metodi richiedono un intero opzionale di stile. Se omesso, viene usato il valore *DateFormat.DEFAULT*, che indica la tipologia di data/ora visualizzata solitamente dal sistema. Altrimenti potete scegliere tra una delle opzioni della **Tabella 1**. Anche qui, un breve esempio per chiarire le idee

```
public class I18nDates {
  public static void main(String[] args) {
    Date currency = new Date(105,2,23,0,40);
    DateFormat dateFormatter;
    String dateOut;
    dateFormatter = DateFormat.getDateTimeInstance(
                DateFormat.FULL, DateFormat.LONG);
    dateOut = dateFormatter.format(currency);
    System.out.println(dateOut);}
}
```

Questo codice lo vedete eseguito nelle **Figure 7** ed **8**: nella prima con *locale en_US*, nella seconda invece la *locale* è *de_DE* (tedesco della Germa-

```
Console
<terminated> I18nDates [Java Application] C:\Programmi\Java\j2sdk1.4.2_03\bin\javaw.exe
Wednesday, March 23, 2005 12:40:00 AM CET
```

Fig. 7: ***Una data in America***

nia). Infine, prima di passare ad Eclipse, un breve cenno alle due classi *MessageFormat* e *ChoiceFormat*.

```
Console
<terminated> I18nDates [Java Application] C:\Programmi\Java\j2sdk1.4.2_03\bin\javaw.exe
Mittwoch, 23. März 2005 00:40:00 CET
```

Fig. 8: ***La stessa data in Germania***

Esse vengono in aiuto del programmatore in quei casi in cui le stringhe da tradurre contengano dei valori determinati a run-time e che quindi devono essere passati come parametro (*MessageFormat*) o, ancora peggio, quando in base al valore del parametro può cambiare una porzione del testo o l'inflessione grammaticale di taluni termini dentro la stringa (*ChoiceFormat*). Per questi argomenti, oltre all'API ufficiale, potete fare riferimento al Java Tutorial citato in una box qui a lato (la sezione specifica si trova all'indirizzo *http://java.sun.com/docs/books/tutorial/i18n/format/messageintro.html*).

ECLIPSE E I18N

Se utilizzate Eclipse per lo sviluppo dei vostri applicativi in Java, avete a disposizione un potente strumento per adattare le vostre applicazioni alla localizzazione rapida. Si tratta del concetto di "esternalizzazione" delle stringhe, che consiste nella ricerca di tutte le stringhe utilizzate nel vostro codice per riportarle su file esterni associandole a delle chiavi di recupero. In questo modo, avrete certamente capito, arriviamo nella situazione dei resource bundle descritta all'inizio. Per vedere in pratica il funzionamento di questa esternalizzazione adopereremo due form già pronti, che non erano stati pensati per l'I18N. Si tratta di *MainWin* e *PropsWin* del progetto di generazione PDF che riprenderemo nel prossimo numero. Selezioniamo il progetto *PdfGeneratorWirhEclipse* e dal menu clicchiamo su *Source->Find Strings to Externalize…*, dovremmo così ottenere una finestra che riepiloga, per classe, il numero di stringhe trovate. Selezionate una classe e cliccate il pulsante *Externalize…*

Si apre la finestra di **Figura 9**, dove potere creare il vostro file di proprietà con tutte le stringhe indicate, modificando il nome delle chiavi se non vi piace e deselezionando le stringhe da non esternalizzare.

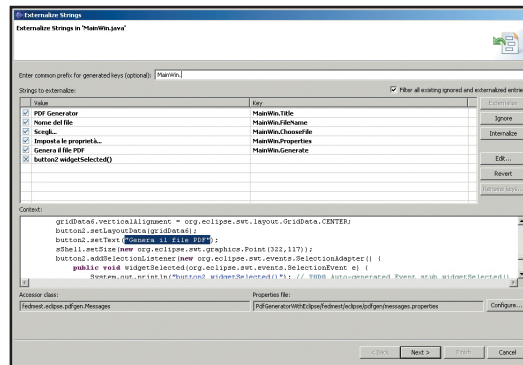



Fig. 9: ***L'interfaccia della procedura di localizzazione assistita in Eclipse***

Cliccando sul pulsante *Configure…* in basso nella finestra potete anche specificare alcune opzioni relative alla classe che sarà usata per recuperare le stringhe e al file di proprietà che sarà creato. Premete poi *Next* due volte e *Finish*. Ripetete l'operazione con l'altra classe.

Alla fine, per fare un esempio, il codice che prima era semplicemente

```
button1.setText("Imposta le proprietà...");
```

dopo l'esternalizzazione diventa

```
button1.setText(NlsMessages.getString(
            "MainWin.PdfProperties")); //$NON-NLS-1$
```

con un file di proprietà in cui troviamo

```
MainWin.PdfProperties=Imposta le proprietà...
```

Il commento che trovate a lato del nuovo codice *($NON-NLS-1$)* sta solo ad indicare che la prima stringa della riga non è da esternalizzare. Se scrivete del codice già pensando di voler poi esternalizzare le stringhe, potete usare quel commento sulle righe le cui stringhe non fanno effettivamente parte del processo di esternalizzazione.

```
Fine – The end – Fin – Fim
```

Se seguite i semplici passi di cui sopra per le vostre applicazioni Java scritte con Eclipse, alla fine della procedura guidata appena vista, per localizzare il software sarà sufficiente aggiungere dei file di proprietà con le traduzioni delle varie chiavi e il giusto suffisso di *locale*.

Ovviamente resta a carico vostro il pensare bene a tutte quelle parti che richiedono flessibilità per adattarsi a culture diverse: abbiamo già parlato di stringhe con parametri, numeri, date, valute, e di come gestirli in maniera *locale*-safe.

Adesso è l'ora della pratica!

Federico Mestrone

LOCALE E STANDARD ISO

Una *locale* è definita da due codici di due lettere ciascuno. Il primo rappresenta la lingua ed è dato dallo standard ISO-639. Trovate un elenco completo di questi codici (due lettere minuscole) all'indirizzo www.ics.uci.edu/pub/ietf/http/related/iso639.txt. Il secondo indica invece la nazione e ci permette di identificare la variante regionale della lingua scelta. Si tratta di due lettere maiuscole il cui significato è dato dallo standard ISO-3166, e ne trovate un elenco completo all'indirizzo www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html.

Scrivere codice in Visual Basic .NET 2003

Le variabili ed i tipi di dati

In questo appuntamento inizieremo a scrivere del codice VB introducendo i mattoncini fondamentali della programmazione: le variabili ed i tipi di dati

Le informazioni che vengono inviate ad un programma dall'esterno, o che vengono utilizzate al suo interno, sono definite dati. Tutti i programmi ricevono dei dati, li utilizzano e li rimandano all'esterno. Il mezzo a disposizione del nostro programma per recuperare o elaborare i dati, è rappresentato dalle variabili. Le variabili sono aree di memoria, battezzate dal programmatore, nelle quali sono temporaneamente archiviati i dati, e sono identificate all'interno dell'applicazione, da un nome e da un tipo di dati che ne definisce il tipo di informazioni memorizzate. I dati utilizzati più frequentemente sono numeri e stringhe. I numeri possono essere positivi, negativi, numeri interi, decimali oppure qualsiasi altro numero che possiamo immaginare. Le stringhe sono un insieme di caratteri. Un carattere è rappresentato da qualsiasi cosa sia digitata mediante la tastiera, tra cui consonanti, vocali, simboli di punteggiatura e, udite udite, anche i numeri. Un singolo carattere è considerato una stringa così come una frase completa.

DICHIARAZIONE DELLE VARIABILI

In VB.NET 2003 è molto semplice dichiarare una variabile, è infatti sufficiente, usare la seguente sintassi:

```
Dim MiaVariabile As TipoDati
```

Da ciò è evidente che gli elementi essenziali da stabilire nella dichiarazione di una variabile sono:

- Il nome della variabile.
- Il tipo di dati che possono essere contenuti nella variabile.

Ci sono almeno tre validi motivi per cui si deve definire il tipo di dati per tutte le variabili:

- Necessità di riconoscere facilmente nel codice il tipo di dati memorizzabili in ogni variabile.
- Utilizzazione ottimizzata della memoria, poiché alcuni tipi di dati occupano più memoria di altri.
- Evitare che una variabile contenga un tipo di dato diverso da quello che ci si aspetta, causando così un errore di non facile individuazione.

Se, ad esempio, vogliamo dichiarare una variabile che dovrà contenere un valore numerico intero, dobbiamo scrivere:

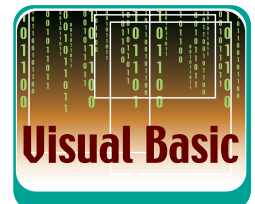
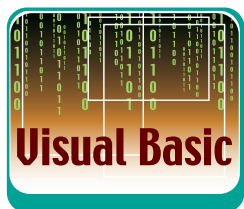
```
Dim PrimaVariabile as Integer
```

In VB.NET è, inoltre, possibile dichiarare più variabili sulla stessa riga di codice, per questo se dobbiamo dichiarare due variabili di tipo intero, possiamo scrivere:

```
Dim Variabile1, Variabile2 As Integer
```

NOMI DI VARIABILE

Nell'assegnazione dei nomi delle variabili, VB.NET impone le poche regole riportate nella Tabella in ultima pagina. Osservando queste regole è possibile attribuire qualsiasi nome ad una variabile, l'unico limite è la nostra fantasia. Nonostante tutto ecco alcuni consigli:

- Per la leggibilità del codice è consigliabile attribuire nomi significativi alle variabili, se si deve assegnare un valore ad un totale di fattura, è consigliabile chiamare la variabile proprio *TotaleFattura*, piuttosto che *T* oppure pippo.
- Non esageriamo con le abbreviazioni dei nomi, è più facile ricordare perché abbiamo utilizzato la variabile *TotaleFattura* piuttosto che la variabile *Tot*.

Utilizza questo spazio per le tue annotazioni

Per disegnare un controllo Button sulla Form si deve:

• Cliccare sull'icona Button presente nella barra degli strumenti.

• Spostare il puntatore sulla form. Noterete che il puntatore assume la forma del mirino con accanto l'icona del componente selezionato.

• Tenere premuto il pulsante del mouse nel punto in cui si desidera collocare l'angolo superiore sinistro dell'etichetta e trascinare il puntatore nel punto in cui si desidera collocare l'angolo inferiore destro.

INIZIALIZZATORI

In VB.NET è possibile dichiarare ed inizializzare una variabile all'interno della stessa istruzione. Questa funzionalità consente di semplificare il codice. Ad esempio si può scrivere:

```
Dim Anno As Integer = 2004
Dim Nome As String = "Nino"
```

Non è, invece possibile, inizializzare più di una variabile dichiarata nella stessa istruzione *Dim, Public*, o *Private* a meno che siano presenti più clausole *As*:

```
'istruzione Scorretta
Dim a, b As Integer = 10
'istruzione corretta
Dim a As Integer = 10, b
As Integer = 10
```

- Se vogliamo identificare una variabile con un nome composto, evitiamo il carattere di sottolineatura *(_)*, ma distinguiamo i termini digitando in maiuscolo la prima lettera di ogni singola parola (*TotaleFattura* piuttosto che *Totale_Fattura*)
- VB è un linguaggio *Case-Insensitive*, cioè non fa distinzione tra i caratteri maiuscoli e minuscoli, perciò *TotaleFattura* e *totalefattura* rappresentano la stessa variabile. Se dichiariamo la variabile come *TotaleFattura*, ogni volta che utilizzeremo questa variabile all'interno del codice, indipendentemente da come sarà scritta (tutta minuscola oppure tutta maiuscola, ecc), sarà trasformata sempre con la stessa combinazione di caratteri minuscolo/maiuscolo con cui l'abbiamo dichiarata. Se usiamo sempre questa convenzione avremo un immediato riscontro visivo sulla corretta scrittura della variabile.

ATTRIBUIRE UN VALORE AD UNA VARIABILE

Per attribuire un valore ad una variabile è sufficiente usare il segno = (uguale), in particolare per assegnare un valore numerico ad una variabile è sufficiente scrivere:

```
TotaleFattura = 1000
```

Le variabili possono contenere soltanto un valore per volta, per cui se scriviamo due istruzioni

```
TotaleFattura = 1000
TotaleFattura = 3000
```

VB pone, in un primo momento, *TotaleFattura* pari a *1000* e successivamente pone *TotaleFattura* pari a *3000* senza memorizzare da nessuna parte che in precedenza il suo valore era pari a *1000.* Ad una variabile è possibile, inoltre, assegnare il valore di una qualsiasi espressione:

```
TotaleFattura=1000 * 20 / 100
```

Per assegnare un valore di tipo stringa è sufficiente racchiudere la stringa tra virgolette:

```
MioNome = "Luigi"
```

VB.NET prevede delle scorciatoie nel momento in cui si esegue un'operazione matematica (o con stringhe) su una variabile, e si vuole memorizzare il risultato all'interno della variabile stessa.
Ad esempio si può scrivere:

```
Dim x As Double = 10.0
x += 1 'Incrementa x di uno (equivale a  x = x + 1)
x -= 3 'Decrementa x di tre (equivale a  x = x - 3)
 x *= 4 'Moltiplica x per quattro (equivale a x = x * 4)
 x /= 4 'Divide x per quattro (equivale a x = x / 4)
'per aggiungere un valore ad una stringa si utilizza
'l'operatore di concatenamento &
Dim s As String
s &= "ABC" 'equivale a s = s & "ABC"
```

VISIBILITÀ E DURATA DELLE VARIABILI

Le variabili in VB.NET non sono tutte uguali, alcune hanno una vita uguale alla vita dell'applicazione mentre altre sono create e distrutte continuamente. L'Area di visibilità (*Scope*) determina l'accessibilità di una variabile all'interno dell'applicazione. La durata (*lifetime*) di una variabile è il periodo per il quale tale variabile resta attiva ed utilizza memoria. Ad esempio, una variabile definita all'interno di una *WindowsForm*, viene creata ogni volta che viene visualizzata la finestra e viene distrutta quando la finestra viene distrutta. VB.NET consente di dichiarare la visibilità delle variabili a livello di:

- **Blocco**
- **Procedura**
- **Modulo**
- **Spazio dei nomi**

Variabili di blocco

Questa caratteristica, consente di dichiarare una variabile all'interno di un blocco di codice come il ciclo *For…Next* ed il costrutto *If..Then..Else* di cui parleremo nei prossimi articoli. Le variabili così dichiarate, presentano l'area di visibilità più ridotta. Hanno visibilità solo all'interno del blocco in cui sono state dichiarate, anche se il ciclo di vita corrisponde con quello della procedura in cui si trovano. Si deve porre particolare attenzione a questa caratteristica, se il flusso di codice rientra più volte nel blocco, senza però uscire dalla procedura in cui il blocco è contenuto, la variabile non viene reinizializzata e contiene il valore che aveva all'uscita del blocco stesso. In alcuni casi potrebbe essere necessario inizializzare la variabile all'inizio del blocco per assicurarci che il programma non utilizzi incidentalmente un valore errato. Questo tipo di dichiarazione migliora la leggibilità del codice, poiché permette di mantenere la dichiarazione vicina al codice che la usa rendendo più facile individuare i punti in cui vengono utilizzate. Nella dichiarazione di variabili di blocco si devono osservare queste regole:

- Non è possibile dichiarare una variabile con lo stesso nome sia a livello di procedura sia a livello di blocco, poiché in questo caso il compilato-

re genera un errore. Il codice seguente genera il seguente errore:

La variabile "x" nasconde una variabile in un blocco di inclusione

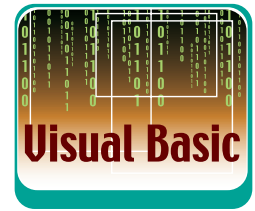
```
Dim indice As Integer
    Dim x As Integer
        For indice = 1 To 5
                Dim x As Integer
            'istruzioni del ciclo
    Next
```

- È possibile dichiarare una variabile di blocco con lo stesso nome di un'altra variabile a livello di classe o globale.
- È possibile dichiarare più variabili di blocco con lo stesso nome nella stessa procedura, a patto che appartengano a blocchi distinti non annidati.

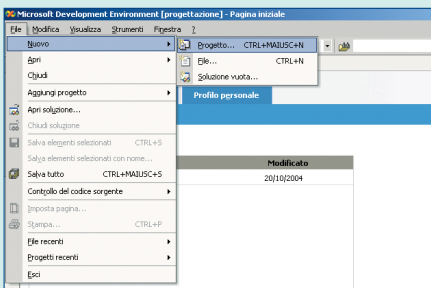
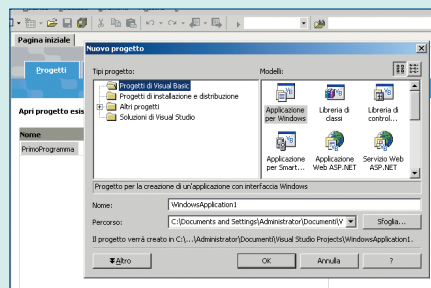
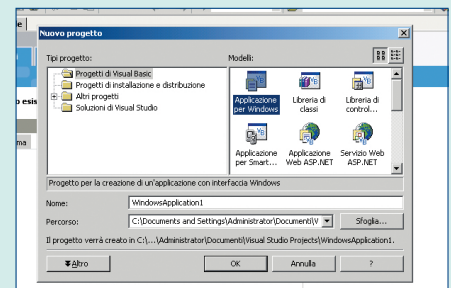
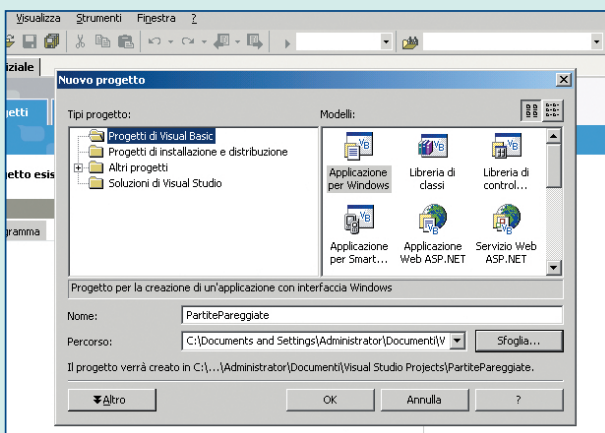
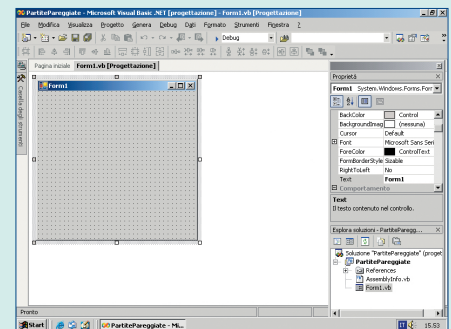
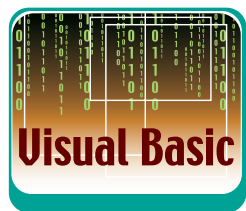
Variabili di procedura dinamiche e statiche

Le variabili locali esistono, e possono essere utilizzate, soltanto all'interno della procedura in cui vengono dichiarate. Una procedura è un blocco di codice autonomo ed è di fondamentale utilizzo poiché: consente di suddividere un programma in tanti sotto-programmi rendendolo più comprensibile, promuove il riutilizzo del codice. Le variabili locali possono essere dichiarate utilizzando l'istruzione *Dim* oppure *Static:*

- se si usa l'istruzione *Static* il valore della variabile rimane in memoria per l'intera durata del modulo che la contiene
- Se si usa l'istruzione *Dim* la variabile esiste solo durante l'esecuzione della procedura in cui è stata dichiarata

È buona norma isolare una variabile nella singola routine in cui viene utilizzata, in modo da poter identificare facilmente errori derivanti da un uso scorretto di tale variabile. È possibile creare più procedure contenenti la dichiarazione di una variabile con lo stesso nome, ciascuna procedura allocherà uno spazio diverso in memoria e riconoscerà la pro-

Per attribuire un valore alle proprietà di un oggetto vale lo stesso discorso fatto per le variabili, perciò per cambiare il contenuto della proprietà *Text* del bottone *ButtonPartite*, all'interno della finestra del codice invece che dalla finestra delle proprietà, possiamo scrivere: *ButtonPartite.Text = "Visualizza la media delle partite pareggiate"*

CREA UNA NUOVA APPLICAZIONE

Per riassumere quanto abbiamo appreso, possiamo creare un programma che calcoli la media delle partite pareggiate dall'Inter nel campionato di Calcio di Serie A. Per iniziare a scrivere l'applicazione, dobbiamo creare un nuovo progetto con i seguenti passi:

1 **Clicchiamo sul pulsante *Nuovo progetto* della barra degli strumenti, oppure selezioniamo la voce di menu *File/Nuovo/Progetto.***

2 **Nella finestra *Nuovo progetto* selezioniamo l'opzione *Progetti di Visual Basic* nella struttura ad albero della casella *Tipi progetto.***

3 **Nella casella *Modelli* selezioniamo l'opzione *Applicazione per Windows.* È chiaro che stiamo sviluppando un'applicazione Standalone.**

4 **Nella casella di testo *Nome* digitiamo il nome del progetto: *PartitePareggiate*. Se è necessario salvare il progetto in un percorso diverso da quello predefinito, si deve immettere il percorso nella casella Percorso (oppure premendo il tasto *Sfoglia* si può navigare tra le directory per selezionare quella desiderata).**

5 **Clicchiamo sul tasto *OK*. In questo modo si apre la finestra di disegno con una finestra (form) vuota denominata *Form1.VB*.**

Per la corretta assegnazione di un nome ad una variabile si devono seguire alcune regole, in particolare il nome di una variabile:

- **Deve iniziare con una lettera dell'alfabeto o con un carattere di sottolineatura (_)**
- **Se inizia con un carattere di sottolineatura deve contenere almeno un carattere alfabetico o un numero**
- **Deve contenere solo caratteri alfabetici, numeri e caratteri di sottolineatura (_). In particolare non è ammesso usare gli spazi ed i simboli di punteggiatura**
- **Non può essere composto da più di 1023 caratteri**

pria variabile.

Variabili a livello di modulo

Le variabili a livello di modulo, sia esso un generico modulo, un modulo *WindowsForm* oppure un modulo di classe, possono essere utilizzate soltanto dalle altre routine presenti nello stesso modulo. Per dichiarare una variabile a livello di modulo è sufficiente usare la parola chiave *Dim* nella sezione dichiarazioni del modulo. Per evidenziare che la variabile può essere usata solo all'interno del modulo, è consigliabile, per la sua dichiarazione, usare la parola chiave *Private*. Le variabili dichiarate con la parola chiave Private, dispongono di accesso privato e sono accessibili soltanto all'interno del contesto in cui vengono dichiarate. Utilizzando la parola chiave *Private* è possibile omettere la parola chiave *Dim*

```
Private VariabileDiModulo As Integer
```

L'area di memoria necessaria a memorizzare i valori di una variabile di modulo, viene allocata nel momento in cui il modulo viene caricato in memoria e viene rilasciata quando il modulo viene scaricato completamente. La vita di una variabile di modulo coincide pertanto con la vita del modulo stesso.

Variabili Spazio dei nomi

Le variabili *Spazio* dei nomi (chiamate comunemente variabili globali) sono le variabili concettualmente più semplici e comode da utilizzare, poiché possono essere utilizzate e modificate in qualsiasi punto del programma. La durata di tali variabili è pari alla vita dell'applicazione. In generale non è consigliabile fare un uso eccessivo delle variabili globali, poiché rendono difficili:

- la manutenzione del codice;
- il riutilizzo del codice;
- la comprensione della logica dell'applicazione.

Fig. 1: ***Le proprietà del bottone***

Proviamo a pensare cosa succede se in un qualsiasi punto del programma si fa un uso scorretto del valore di una variabile globale, in questo caso saremo costretti a rileggere l'intero codice per individuare l'errore. In VB.NET Le variabili globali sono dichiarate utilizzando, in un modulo, la parola chiave *Public* al posto del comando *Dim:*

```
Public VariabileGlobale As Integer
```

TORNIAMO ALL'APPLICAZIONE D'ESEMPIO

Sulla finestra vuota, possiamo disegnare un controllo *Button*, così come abbiamo visto nel precedente articolo (vedi box, per i più distratti) e modificarne il nome ed il testo visualizzato. Dalla finestra delle proprietà cambiamo la proprietà *Name* in *ButtonPartite* e la proprietà *Text* in *Visualizza* la media delle partite pareggiate. A questo punto, possiamo scrivere il codice all'interno della procedura di evento *ButtonPartite_Click* che viene eseguito ogni qualvolta l'utente clicca con il mouse sul pulsante. Ricordiamo che per aprire la finestra del codice, con il cursore posizionato all'interno della procedura di evento *ButtonPartite_Click*, è sufficiente fare doppio clic sul bottone disegnato sulla form.

```
Private Sub ButtonPartite_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        Handles ButtonPartite.Click
    Dim PartitePareggiate As Integer
    Dim TotalePartite As Integer
    Dim PercentualePareggi As Single
    PartitePareggiate = 9
    TotalePartite = 11
    PercentualePareggi = PartitePareggiate
                                    / TotalePartite
    MessageBox.Show(PercentualePareggi)
  End Sub
```

Ecco come Visual Basic.NET interpreta questi sette comandi:

1. Il primo comando crea una variabile denominata: *PartitePareggiate* di tipo numerico intero;
2. Il secondo comando crea una variabile denominata: *TotalePartite* di tipo numerico intero;
3. Il terzo comando crea una variabile denominata: *PercentualePareggi* di tipo numerico decimale;
4. Il quarto comando imposta il valore della variabile *PartitePareggiate* pari a 9;
5. Il quinto comando imposta il valore della variabile *TotalePartite* pari a 11;
6. Il sesto comando imposta il valore della variabile *PercentualePareggi* pari al valore di *PartitePa-*

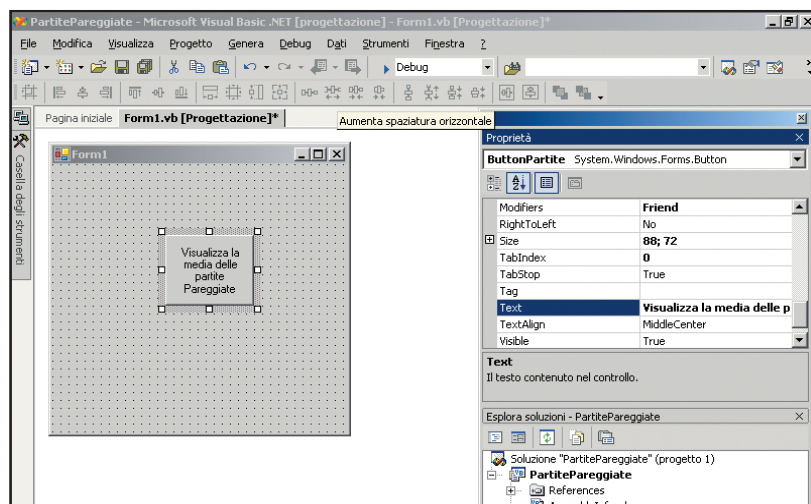
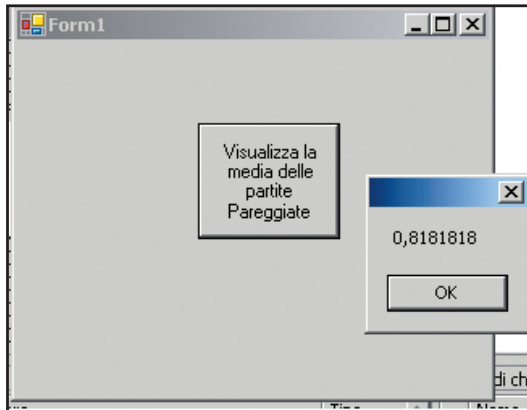



Fig. 2: L'applicazione in esecuzione

reggiate diviso il valore di *TotalePartite* (tramite l'operatore /), in modo da calcolare la media delle partite pareggiate. In questo caso il valore di *PercentualePareggi* è uguale ad 8/11, ovvero 0,818181;

7. Il settimo ed ultimo comando mostra un messaggio a video, con il valore della variabile *PercentualePareggi* (Tramite il metodo *Show* della classe *MessageBox* introdotta nel precedente articolo).

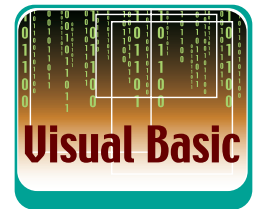
TIPI DI DATI

Il tipo di dati di una variabile determina il tipo di informazione e, di conseguenza, il numero di bit necessari alla memorizzazione dei valori nel computer. Definendo il tipo di dati di una variabile si possono evitare equivoci, se infatti, si scrive una istruzione che tenta di assegnare una stringa ad una variabile di tipo numerico, VB visualizza un messaggio di errore. In questo modo diventa più semplice identificare eventuali errori presenti nel programma. Fino a questo punto abbiamo utilizzato le variabili per memorizzare valori numerici e stringhe, tuttavia numeri e stringhe non sono le uniche informazioni che è possibile memorizzare, ma esistono anche altri tipi di dati riassunti nella tabella seguente.

CONCLUSIONI

Nel primo articolo ci siamo soffermati sull'aspetto e l'utilizzo dell'ambiente di sviluppo di VB .NET 2003, in questo secondo appuntamento, ci siamo avvicinati al linguaggio vero e proprio, esaminando l'utilizzo delle variabili ed elencando i tipi di dati supportati dal compilatore.

Nei prossimi articoli ci addentreremo sempre di più nei meandri della programmazione VB. NET

Luigi Buono

NOTA

Esistono due tipi di variabili, quelle definite dal programmatore e quelle definite da VB .NET come proprietà di ciascun oggetto presente sulla form. In pratica ogni volta che si crea una nuova finestra, e che viene disegnato un oggetto per comporre l'interfaccia utente, VB crea in automatico le proprietà dell'oggetto disegnato, impostandole ai valori di default.

I TIPI DI DATI ACCETTATI DA VB

Tipo di dati	Accetta i valori	Occupa in memoria	Nota Bene
Byte	numerici interi (non è possibile usare il punto decimale) compresi tra zero e 255	un Byte (8 bit)	*è il tipo più piccolo consentito in VB, ed è di scarsa utilità pratica*
Short	numerici interi tra –32.768 e 32.767	due byte	
Integer	numerici interi tra –2.147.483.648 e 2.147.483.647	quattro byte	*in ambienti a 32-bit il tipo Integer è più efficiente rispetto al tipo Long*
Long	numerici interi tra –9.223.372.036.854.775.808 e 9.223.372.036.854.775.807	otto byte (64 bit)	
Single	numerici decimali tra –3,402823E38 e –1,401298E-45 per valori negativi e tra 1,401298E-45 e –3,402823E38 per valori positivi	quattro byte	
Double	numerici decimali compresi tra –1,79769313486232E308 e –4,94065645841247E-324 per i valori negativi e tra 4,94065645841247E-324 e 1,79769313486232E308 per i valori positivi	otto byte	*Nella maggior parte dei casi è il tipo più adatto, per manipolare dei valori numerici decimali*
Decimal	numerici decimali che possono contenere fino a 28 cifre a destra della virgola	sedici byte	*è il tipo più indicato Nei calcoli di natura finanziaria, perché permette di evitare problemi di arrotondamento e di troncamento*
Boolean	Possono essere soltanto *True* o *False*	due byte	*Per assegnare un valore si devono utilizzare le parole chiave True e False*
String	può contenere da zero a circa due miliardi di caratteri Unicode	per ogni carattere richiede uno spazio spazio in memoria pari a due byte.	*è il tipo da utilizzare per memorizzare memorizzare parole e lettere*
Char	Può contenere un solo carattere Unicode	due byte	*Quando si deve utilizzare una variabile che debba contenere un solo carattere, l'utilizzo del tipo Char rispetto al tipo String è più conveniente poiché consente migliori prestazioni*
Date	una data qualsiasi compresa tra il 1° Gennaio 0001 ed il 31 Dicembre 9999, e possono inoltre contenere un qualsiasi valore di orario	otto byte	
Object	può essere assegnato un qualunque tipo	vengono memorizzate come indirizzi di 32 bit (4 byte) riferiti a oggetti	

Come usare gli array di oggetti e le collezioni di Java

Collezionare oggetti

La gran parte dei programmi devono creare migliaia o milioni di oggetti. Ma naturalmente non puoi definire milioni di variabili. Questo mese imparerai come conservare e recuperare gli oggetti

CD WEB
codice_corso_java_18.zip
cdrom.ioprogrammo.it

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
Array di tipi primitivi, ereditarietà, polimorfismo

Software
Java 2 Standard Edition SDK 1.4 o superiore

Impegno

Tempo di realizzazione

Il capo della tua azienda, manager illuminato, ti ha chiesto un programma per gestire il club aziendale di calcetto. Tra le altre funzionalità, il programma deve stampare sullo schermo un elenco di tutti i soci.

```
public class Socio {
    private final String titolo;
    private final String nome;
    private final String cognome;
    public Socio(String titolo, String nome, String
                                              cognome) {
        this.titolo = titolo;
        this.nome = nome;
        this.cognome = cognome; }
    public String getBigliettoDaVisita() {
        return titolo + " " + nome + " " + cognome;}
}
```

Un *Socio* è una semplice struttura dati, con un titolo (come "Sig.", o "Dott."), un nome e un cognome. Il metodo *getBigliettoDaVisita()* restituisce una descrizione del socio sotto forma di stringa.

Esercizio 1: Scrivi un programma che costruisce una breve lista di Soci e li passa ad un metodo *stampaSoci()*. Questo metodo riceve un elenco di Soci di lunghezza qualsiasi e stampa sullo schermo i biglietti da visita di tutti i soci nell'elenco.

Il problema è costruire l'elenco. Puoi risolvere questo esercizio con un array. Ma, se non hai mai provato a costruire un array di oggetti, la cosa potrebbe rivelarsi più difficile del previsto. Ecco una soluzione:

```
public class ClubConArray {
    public static void main(String[] args) {
        Socio[] soci = new Socio[3];
        soci[0] = new Socio("Ing.", "Piero", "Serbelloni");
        soci[1] = new Socio("Cav.", "Giangigi", "Mazzanti");
        soci[2] = new Socio("Lup. Mann.", "Pierpiero",
                                        "Viendalmare");
        stampaSoci(soci); }
    private static void stampaSoci(Socio[] soci) {
        for(int i = 0; i < soci.length; i++)
        System.out.println(soci[i].getBigliettoDaVisita()); }
}
```

ClubConArray.stampaSoci() riceve un array di soci, attraversa l'array con un ciclo *for* (qualunque sia la sua lunghezza) e stampa il biglietto da visita di ciascun socio. Il *main()* costruisce un array di tre soci e lo passa alla funzione.

TANTI OGGETTI, TUTTI IN FILA

Soffermiamoci sull'array di Soci. Se avessimo un array di variabili primitive, ad esempio di *int*, tutto sarebbe molto semplice:

```
int[] arrayDiPrimitive = new primitive[3];
arrayDiPrimitive[0] = 12;
```

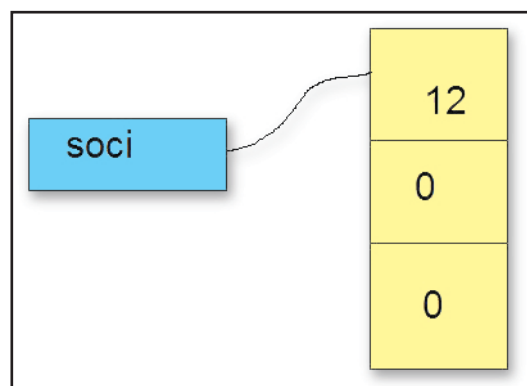
Questo array contiene tre *int*, il primo dei quali vale *12*. Ma anche gli altri componenti hanno un valore: quello di default, che per gli int è *0*. Puoi vedere la situazione in **Figura 1**: il reference *arrayDiPrimitive* punta ad una sequenza di tre variabili *int*.
Gli array di oggetti sono un po' più delicati. Nel nostro esempio, l'array *soci* contiene tre reference ad oggetti di tipo *Socio*, indicizzati da *0* a *2*. Qual è il valore iniziale di questi reference? Il valore di default di un reference non inizializzato è *null*, cioè "nessun

soci
12
0
0

Fig. 1: Una rappresentazione grafica dell'array Soci

oggetto" (abbiamo parlato del valore *null* il mese scorso). Se qualcuno cerca di eseguire un metodo su un *reference null,* il sistema genera una *NullPointerException.* Quindi è bene che i reference *null* siano inizializzati il prima possibile.
Riassumendo: quando crei un array di oggetti, ricorda di fare due cose:

1 creare l'array (con *new*);
2 creare, il prima possibile, tutti gli oggetti (anche in questo caso devi usare *new*).

Il primo passo è identico a quello che già conosci per le variabili primitive:

```
Socio[] soci = new Socio[3];
```

Il secondo passo vale anche per gli array di primitive, ma è più importante per gli array di oggetti. Le prime volte che programmavo in Java, mi è capitato di dimenticarlo. Da allora ho imparato, e il programma di questo mese inizializza subito gli elementi dell'array:

```
soci[0] = new Socio("Ing.", "Piero", "Serbelloni");
```

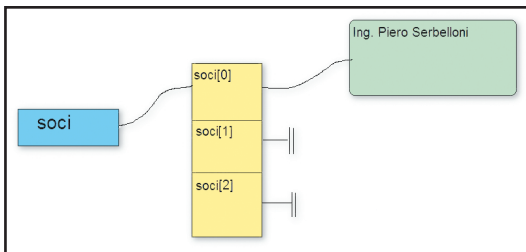



Fig. 2: ***Una "fotografia" della struttura dei dati durante l'inizializzazione***

La **Figura 2** mostra la situazione della memoria subito dopo che è stata eseguita questa riga: un reference di nome *soci* punta all'array, che contiene tre reference, il primo dei quali punta ad un Socio. Gli altri due reference sono *null*. Per fortuna le due righe successive del nostro programma completano la costruzione dell'array.

IL MONDO REALE

Molte cose cambierebbero se *ClubConArray* fosse un vero programma di gestione di club. La classe *Socio* non sarebbe più un semplice contenitore di dati, ma acquisterebbe metodi come *pagaQuotaDiPartecipazione()* e *iscrivitiAPartita();* i soci sarebbero forse conservati in un database; l'utente potrebbe usare un'interfaccia grafica per inserire nuovi soci. Quest'ultimo requisito è più complicato di quello che potrebbe sembrare. Se l'utente del programma registra un nuovo socio, il programma deve aggiungerlo all'array soci. Qui nasce un dilemma. Quando crei un array, devi immediatamente dire quanto sarà grande, in modo che il sistema possa riservare (allocare) una fetta di memoria con le dimensioni giuste per contenere tutti i reference. Ma nel momento in cui scrivi il programma, non sai quanti soci si iscriveranno al club. Il loro numero potrebbe cambiare in qualsiasi momento durante l'esecuzione del programma. Allora, quali dimensioni devi dare all'array? Per dirla in termini più tecnici, il problema degli array è che sono statici: le dimensioni di un array sono definite all'inizio e non possono più cambiare. Quello che ci servirebbe è un contenitore dinamico, che si espande man mano che gli vengono aggiunti elementi e si riduce quando gli elementi vengono rimossi. Per fortuna le librerie di Java ti mettono a disposizione alcuni di questi contenitori: si chiamano collezioni.

PRONTI A CAMBIARE

Per usare le collezioni di Java devi importare il package *java.util*:

```
import java.util.*;
```

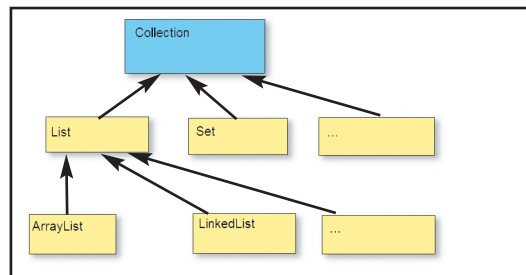



Fig. 3: ***La gerarchia di Collection***

Tutte le collezioni implementano l'interfaccia *java.util.Collection.* Le collezioni più semplici sono le liste, semplici elenchi ordinati di oggetti molto simili agli array. Tutte le liste derivano dall'interfaccia *java.util.List* che, a sua volta, è una sotto-interfaccia di Collection (**Figura 3**). Negli esempi di questo mese uso una particolare implementazione concreta di *List* che si chiama *ArrayList*. L'interfaccia *List* definisce un metodo **add()** per aggiungere un oggetto, un metodo *remove()* per rimuovere un oggetto, e un metodo *get()* per prelevare un oggetto a partire dal suo indice. Un esempio:

```
ArrayList soci = new ArrayList();
Socio s = new Socio("Ing.", "Piero", "Serbelloni");
soci.add(s);
s.add(new Socio("Cav.", "Giangigi", "Mazzanti"));
System.out.println("Il club ha " + soci.size() + " soci");
```

Collection.size() restituisce il numero di elementi in una collezione.

OBIETTIVI

Questo mese:

- **Costruirai i tuoi primi array di oggetti.**
- **Scoprirai pregi e difetti delle collezioni.**
- **Conoscerai Object, la madre di tutte le classi.**

LEGGERE E SCRIVERE

Il nome *Socio.getBigliettoDaVisita()* mescola inglese e italiano in modo piuttosto goffo, ma voglio rispettare la convenzione di Java per la quale i metodi che leggono una proprietà iniziano con get e quelli che la scrivono iniziano con set. Per evitare nomi ridicoli come questo è bene usare solo l'inglese (*getBusinessCard()* suona sicuramente meglio), ma in questo corso cercherò di essere meno esterofilo.

Esercizio 3: Crea un'ArrayList e inseriscici qualche oggetto (Soci, o stringhe, o quello che ti pare). Alla fine usa il metodo *size()* per recuperare il numero di elementi nella lista. Rimuovi uno o più oggetti con il metodo *remove()* e leggi nuovamente la lunghezza della lista.

Per recuperare un elemento dalla lista puoi usare il metodo *get()*. Questo metodo richiede un indice simile a quello di un array, che varia da 0 alla lunghezza della lista meno uno:

```
for(int i = 0; i < soci.size() - 1; i++)
    System.out.println(soci.get(i));
```

Ecco il nostro club implementato con una lista:

```
public class ClubConLista {
    public static void main(String[] args) {
        ArrayList soci = new ArrayList();
        soci.add(new Socio("Ing.", "Piero", "Serbelloni"));
        soci.add(new Socio("Cav.", "Giangigi", "Mazzanti"));
        soci.add(new Socio("Lup. Mann.", "Pierpiero",
                                            "Viendalmare"));
        stampaSoci(soci); }
    private static void stampaSoci(ArrayList soci) {
        for(int i = 0; i < soci.size(); i++)
            System.out.println(soci.get(i)); }
}
```

Nota che la dimensione della lista non viene mai fissata, ma dipende sempre dal numero di elementi che ci infili dentro. Le liste sono dinamiche, a differenza degli array. Se fai girare il programma, però, avrai una sgradevole sorpresa:

it.ioprogrammo.corsojava.collezioni.Socio@360be0
it.ioprogrammo.corsojava.collezioni.Socio@45a877
it.ioprogrammo.corsojava.collezioni.Socio@1372a1a

Cosa diavolo è questa robaccia? Il problema è che il programma cerca di stampare dei Soci, ma non sa come fare. Quindi stampa, per ciascun *Socio*, alcune informazioni di sistema. Questo è il comportamento di default per gli oggetti Java. Esistono modi eleganti per aggirare il problema, ma per ora seguiremo la strada più semplice: anziché stampare gli oggetti, stamperemo la stringa restituita dal metodo *Socio .getBigliettoDaVisita()*.
Sembra una buona idea. Ma se ci proviamo, succede qualcosa di ancora più brutto:

```
for(int i = 0; i < soci.size(); i++)
   System.out.println(soci.get(i).getBigliettoDaVisita());
                                                  // errore!
```

Esercizio 4: Riesci a capire perché questo codice non compila?

Se ricordi bene come funziona il polimorfismo, forse puoi arrivarci da solo. Ma per spiegare bene cosa succede devo aprire il cofano delle collezioni e mostrarti cosa c'è sotto. Prendiamo come esempio *ArrayList*. Al suo interno, un'*ArrayList* usa un array per contenere gli oggetti. Il vantaggio dell'*ArrayList* sugli array sta nel fatto che questa classe contiene del codice "furbo" che gestisce in modo automatico le dimensioni dell'array. Se aggiungi alla collezione più oggetti di quelli che l'array è in grado di contenere, l'*ArrayList* costruisce un array più grande, ci copia dentro il contenuto del vecchio array e butta via quest'ultimo. Questo è solo uno dei modi possibili per implementare una collezione, e per usare le liste non è necessario saperlo. Dall'esterno vedi semplicemente una lista dinamica, e puoi beatamente ignorare il fatto che questo risultato sia ottenuto manipolando array statici. Ma se provassi tu stesso a scrivere il codice di una classe come *ArrayList*, ti troveresti davanti ad un problema. Quando dichiari un array, devi dargli un tipo:

```
Soci[] soci;
```

Un'*ArrayList*, invece, deve poter contenere non solo dei Soci, ma letteralmente qualsiasi oggetto. Lo stesso problema si pone per tutte le collezioni: a un certo punto gli oggetti che infili nella collezione dovranno essere conservati in qualche reference, e quel reference deve avere un tipo. Ma gli oggetti che vengono inseriti nella collezione possono avere qualsiasi tipo. Quindi, che tipo deve avere il reference? Java 1.5 risolve questo problema con una nuova funzionalità: i generici. Ma Java 1.5 è ancora troppo nuovo e poco usato, quindi non ne parlerò in questo corso. Nelle versioni precedenti di Java il segreto sta in una delle caratteristiche più importanti del linguaggio: tutte le classi, ma proprio tutte, hanno un antenato in comune.
Quest'antenato fa parte del package di default *java.lang*.
Si chiama *Object*. Se prendi tutte le classi di qualsiasi programma Java e disegni un enorme albero che mostra chi deriva da cosa, l'albero avrà sempre un'unica radice, che è la classe *Object*. Non importa se una classe eredita da un'altra classe: se vai sempre più su nella gerarchia, prima o poi troverai un antenato che eredita direttamente da *Object*. Non ci credi? Prova a fare un cast da un oggetto di tipo *Socio* ad un reference di tipo *Object*:

```
Socio s = new Socio();
Object o = s;   // funziona!
```

Il compilatore accetta il cast, anche se non specifico mai che *Socio* deriva da *Object*. Questo perché tutte le volte che definisci una classe senza usare la parola chiave *extends*, Java la aggiunge silenziosamente.

NOTA

IO NON POSSO ENTRARE

E' facile mettere qualsiasi oggetto in una *Collection*. A volte, però, quello che devi conservare non sono oggetti ma variabili primitive. In questo caso devi "avvolgere" la variabile in un oggetto per renderla compatibile con le collezioni, che accettano solo oggetti. Nel package *java.lang* troverai una serie di classi che servono proprio a questo, chiamate wrapper. Se vuoi mettere un char in una collezione puoi creare un oggetto di classe *Character*. *Character* ha un costruttore che prende un char e un metodo *charValue()* che restituisce il char originale. Ciascun tipo primitivo di Java ha una classe wrapper corrispondente.

È come se nella dichiarazione della classe *Socio* avessi scritto:

```
class Socio extends Object {
  // ...
}
```

In termini di polimorfismo, questo significa che qualsiasi oggetto Java è un *Object*. Quindi puoi definire un reference a *Object* e assegnargli qualsiasi oggetto. Questo spiega come faccia *ArrayList* a contenere qualsiasi cosa: all'interno, conserva gli oggetti in un *Object[]*. Se vai a leggere le specifiche dell'interfaccia *java.util.List* vedrai qualcosa di simile:

```
public interface List {
    int size();
    boolean add(Object o);
    Object get(int index);
    // altri metodi...
}
```

Il metodo *add()* prende un *Object* (e restituisce un boolean che indica se l'oggetto era già nella lista). Il metodo *get()* prende un indice intero e restituisce un *Object*. Quindi questo meccanismo funziona con tutti gli oggetti a prescindere dalla loro classe. Questa è una bella cosa, ma comporta un problema: quando estrai un *Socio* dalla lista, quello che ti ritrovi non è un reference a *Socio*, ma un reference a *Object*. E dato che la classe *Object* non ha un metodo *getBigliettoDaVisita()*, il compilatore si rifiuta di farti invocare questo metodo. Questo è un limite delle collezioni di Java: per funzionare con qualunque tipo, devono dimenticare con quale tipo hanno a che fare volta per volta. Se vuoi una collezione "tipata", che cioé contiene solo oggetti di un tipo specifico, devi scrivertela da solo. Per fortuna esiste una soluzione, anche se non particolarmente elegante.

DALL'ALTO IN BASSO

Java fa il cast verso l'alto degli oggetti in modo automatico. Ad esempio, queste due righe di codice fanno la stessa cosa:

```
Object o1 = new Socio();
Object o2 = (Object)new Socio();
```

L'upcast è un'operazione sicura, perché il compilatore può andare a guardare la definizione di una classe e scoprire con esattezza da quali classi eredita e quali interfacce implementa. Se qualcosa non torna, l'errore si verifica durante la compilazione, e sai per certo che non avrai brutte sorprese durante l'esecuzione. Cosa succede se cerchi di chiamare i metodi della classe *Socio* sull'oggetto *o*? Il compilatore te lo impedirà, perché facendo l'upcast hai volutamente "dimenticato" che avevi a che fare con un *Socio* e hai deciso di trattarlo come un *Object*. Ma se vuoi puoi tornare al tipo specifico dell'oggetto con l'operazione opposta all'upcast: il cast verso il basso, o *downcast*:

```
Socio s = (Socio)o;
s.getBigliettoDaVisita(); // ora funziona!
```

In questo caso ho forzato il cast da *Object* a *Socio*, con una sintassi simile a quella che si usa per fare il cast delle variabili primitive. Il tipo tra parentesi dice al compilatore: lascia perdere i tuoi controlli di tipo; garantisco io che questo reference contiene un *Socio*. Il compilatore presume che tu sappia ciò che stai facendo e ti permette di andare avanti. Da questo momento in poi puoi usare il reference *s* per chiamare tutti i metodi della classe *Socio*. Naturalmente devi essere sicuro che la collezione contenga proprio dei *Soci*, e non qualcos'altro. Cosa succede se ti sbagli? Niente di piacevole...

Esercizio 5: Scrivi un programma che fa un cast sbagliato da una classe all'altra. Come si comporta il compilatore? Cosa succede durante l'esecuzione del programma?

Un esempio è il programma *Downcast* che trovi sul CD. Se lo lanci otterrai una *ClassCastException*. Insieme alle *NullPointerException*, le *ClassCastException* sono tra le più comuni e famigerate eccezioni Java. Per questo motivo, quando hai la possibilità di scegliere, dovresti evitare i downcast. Ma c'è almeno un caso in cui non hai scelta: quando recuperi gli oggetti in una collezione. Ed ecco il nostro programma in una versione finalmente funzionante che usa una lista dinamica e un **downcast** in uscita:

```
public class ClubConListCorretto {
  public static void main(String[] args) {
     ArrayList soci = new ArrayList();
     soci.add(new Socio("Ing.", "Piero", "Serbelloni"));
     soci.add(new Socio("Cav.", "Giangigi", "Mazzanti"));
     soci.add(new Socio("Lup. Mann.", "Pierpiero",
                                             "Viendalmare"));
     stampaSoci(soci); }
private static void stampaSoci(ArrayList soci) {
    for(int i = 0; i < soci.size(); i++) {
       Socio prossimoSocio = (Socio)soci.get(i);
       System.out.println(prossimoSocio
                          .getBigliettoDaVisita()); } }}
```

Mentre mediti su tutto ciò, ne approfitto per darti appuntamento al mese venturo. Il nostro corso è in dirittura d'arrivo, quindi non perdere le ultime puntate!

Paolo Perrotta

NOTA

SIMILI MA DIVERSE

Oltre alle liste, il package *java.util* contiene diversi altri tipi di collezioni, tutte nipotine di *java.util.Collection*. I Set, ad esempio, garantiscono che ciascun oggetto compare nella collezione una sola volta. Ma tutti gli esempi di questo mese usano le più semplici *List*, che sono le *Collection* più comuni. *List* è un'interfaccia (Figura 3). Una delle liste più usate è l'*ArrayList*, che internamente conserva gli oggetti in un array. Esiste anche la *LinkedList*, che usa un sistema più complicato (una specie di catena di "contenitori" di oggetti collegati da reference). Le due classi hanno prestazioni diverse: *ArrayList* è ideale per accedere velocemente agli oggetti, *LinkedList* è più veloce nell'inserirli e nel rimuoverli. In ogni caso, se la tua lista non ha decine di migliaia di elementi, puoi probabilmente ignorare la differenza.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

VISUAL BASIC

TRASCINARE UN CONTROLLO A RUNTIME

Lasciare libero l'utente di trascinare un controllo sul form e spostarlo a runtime: è sufficiente copiare il codice che trovate di seguito per realizzare questa utile funzione. Per utilizzare il codice così com'è, sul form deve essere presente un controllo di nome *Picture1*.

```
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias
  "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal
                        wParam As Long, lParam As Any) As Long
Const WM_NCLBUTTONDOWN = &HA1
Const HTCAPTION = 2
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer,
                                        X As Single, Y As Single)
ReleaseCapture
SendMessage Picture1.hwnd, WM_NCLBUTTONDOWN, HTCAPTION, 0
End Sub
```

COME ESPORTARE DATI DA ACCESS A XML

Su un PC che abbia installato Windows XP e Office 2003, è possibile esportare dati da una tabella Access ad un file XML, con una sola riga di codice. Dopo aver selezionato *"Microsoft Access Object Library"* dal menu *references*, potete inserire nel vostro progetto la seguente riga:

```
opencurrentdatabase(<Databasepath>)
ExportXML "Tablename","<XmlFilepath>,"<xsdfilepath>"
closecurrentdatabase
```

COME MANDARE UNA MAIL

Attraverso il codice che segue, è possibile inviare una mail utilizzando il controllo MAPI:

```
Private Sub CmdExit_Click()
   End
End Sub
Private Sub CmdSend_Click()
  On Error GoTo SendErrorMailError
   'sign on system
   MAPSession.UserName = txtUserid.Text
   MAPSession.Password = txtUserpass.Text
   MAPSession.SignOn
   'send e-mail
   MAPMessage.SessionID = MAPSession.SessionID
   MAPMessage.Compose
   MAPMessage.RecipDisplayName = txtname.Text
   MAPMessage.RecipAddress = txtId.Text
   MAPMessage.AddressResolveUI = False
   MAPMessage.MsgSubject = txtSub.Text
   MAPMessage.MsgNoteText = RtMessage.Text
   MAPMessage.Send False
   'sign out
   MAPSession.SignOff
   MsgBox "Message Sent"
   Exit Sub
SendErrorMailError:
   MsgBox "Error " & Format$(Err.Number) & _
      " sending mail." & vbCrLf & _
      Err.Description
   Exit Sub
End Sub
```

LEGGERE E SCRIVERE FILE DI TESTO

Il codice che andiamo a illustrare è una sorta di Wrapper per le funzionalità di lettura e scrittura dei file di testo. Grazie alle due nuove funzioni potremo accedere ai file senza preoccuparci dei dettagli implementativi più noiosi:

```
public Function ReadFile(FileName as string) as string
   Dim i as Integer
   i = FreeFile
   on error GoTo ErrorTrap
   Open FileName for input as #i
   ReadFile = input(LOF(i), i)
   Close #i
   Exit Function
ErrorTrap:
   ReadFile = ""
End Function
public Sub WriteFile(FileName as string, Contents as string)
   Dim i as Integer
```

```
    i = FreeFile
    Open FileName for Output as #i
    print #i, Contents
    Close #i
End Sub
```

Ed ecco come copiare il contenuto di un file in un altro con le nostre funzioni nuove di zecca:

```
Call WriteFile("c:\arrivo.txt", ReadFile("c:\partenza.txt"))
```

CERCARE NELLA LISTBOX DURANTE LA DIGITAZIONE

Una simpatica funzionalità che si occupa di cercare un elemento in una *listBox*, che corrisponda ai criteri immessi in una *textBox*. Per provare il nostro codice è sufficiente avviare VB 6, e iniziare un nuovo progetto EXE standard. Trascinate sulla form una listBox **(List1)** ed una textBox (*Text1*) e copiate il codice illustrato di seguito.

```
option Explicit
'Start a new Standard-EXE project.
'Add a textbox and a listbox control to form 1
'Add the following code to form1:
private Declare Function SendMessage Lib "User32" _
        Alias "SendMessageA" (byval _
        hWnd as Long, _
        byval wMsg as Integer, _
        byval wParam as string, _
        lParam as Any) as Long
Const LB_FINDSTRING = &H18F
private Sub Form_Load()
    With List1
        .Clear
        .AddItem "Ramarro"
        .AddItem "ramundo"
        .AddItem "RAME"
        .AddItem "ROM"
        .AddItem "Roma"
        .AddItem "Romantico"
        .AddItem "Ruspante"
        .AddItem "Casa"
        .AddItem "Castello"
    End With
End Sub
private Sub Text1_Change()
    List1.ListIndex = SendMessage(List1.hWnd, LB_FINDSTRING, _
                  Text1, byval Text1.Text)
End Sub
```

AGGIUNGERE UNA LABEL A RUNTIME

In molti casi può risultare utile aggiungere dei controlli sulla form, durante l'esecuzione del programma. Il codice che mostriamo consente di aggiungere una label, specificandone il testo:

```
Dim objControl As Label
Set objControl = Controls.Add("VB.Label", "lblNewLabel", Me)
With objControl
    .Caption = "Sono nuova"
    .Left = 100
    .Top = 100
    .Visible = True
End With
```

OTTENERE NUMERI ROMANI

Con questa procedura è possibile trasformare numeri interi decimali in numeri romani (rappresentati ovviamente da una stringa). Il range ammissibile è compreso fra 1 e 4999: se si prova a forzare la procedura con numeri che esulano da questo intervallo, la stringa ritornata sarà in notazione decimale. Il parametro opzionale *iStyle* consente di specificare due stili diversi: *standard*, con *iStyle* posto a *-1* (4 = iv, 9 = ix, e così via); oppure *classico*, con *iStyle* posto a *-2* (4 = iiii, 9 = viiii, eccetera).

```
Public Function Roman(ByVal n As Integer, _
        Optional iStyle As Integer = -1) As String
        If n < 1 Or n >= 5000 Then
                Roman = CStr(n)
                Exit Function
        End If
        If iStyle <> -2 Then iStyle = -1
        Dim sRtn As String, i As Integer, x As Integer
        Dim r(1 To 13) As String, v(1 To 13) As Integer
        r(1) = "i": v(1) = 1
        r(2) = "iv": v(2) = 4
        r(3) = "v": v(3) = 5
        r(4) = "ix": v(4) = 9
        r(5) = "x": v(5) = 10
        r(6) = "xl": v(6) = 40
        r(7) = "l": v(7) = 50
        r(8) = "xc": v(8) = 90
        r(9) = "c": v(9) = 100
        r(10) = "cd": v(10) = 400
        r(11) = "d": v(11) = 500
        r(12) = "cm": v(12) = 900
        r(13) = "m": v(13) = 1000
        x = UBound(v)
        sRtn = ""
        Do
                For i = x To LBound(v) Step iStyle
                        If v(i) 7lt;= n Then
                                sRtn = sRtn & r(i)
                                n = n - v(i)
                                x = i
                                Exit For
                        End If
                Next i
        Loop Until n = 0
        Roman = sRtn
End Function
```

L'IMMAGINE IN UN DOCUMENTO WORD

Una procedura che, passando per la clipboard, consente di copiare, all'interno di un documento Word, una qualsiasi immagi-

ne. Nell'esempio che riportiamo, l'immagine da copiare è quella presente nel controllo Picture1.

```
Private Sub Command1_Click()
Dim file_name As String
Dim file_path As String
Dim file_title As String
Dim txt As String
Dim new_txt As String
Dim pos As Integer
Screen.MousePointer = vbHourglass
Command1.Enabled = False
DoEvents
' Apre Word
file_name = txtFilename.Text
file_title = Mid$(file_name, InStrRev(file_name, "\") + 1)
file_path = Left$(file_name, Len(file_name) - Len(file_title))
' Decommentare la riga seguente per visualizzare Word
' WordServer.Visible = True
WordServer.ChangeFileOpenDirectory file_path
WordServer.Documents.Open _
FileName:=file_title, _
ConfirmConversions:=False, _
ReadOnly:=False, _
AddToRecentFiles:=False, _
PasswordDocument:="", _
PasswordTemplate:="", _
Revert:=False, _
WritePasswordDocument:="", _
WritePasswordTemplate:="", _
Format:=wdOpenFormatAuto
WordServer.Selection.GoTo _
What:=wdGoToBookmark, _
Name:="Disclaimer"
WordServer.Selection.Find.ClearFormatting
With WordServer.Selection.Find
.Text = ""
.Replacement.Text = ""
.Forward = True
.Wrap = wdFindContinue
.Format = False
.MatchCase = False
.MatchWholeWord = False
.MatchWildcards = False
.MatchSoundsLike = False
.MatchAllWordForms = False
```

IL TIP DEL MESE

INIZIALIZZARE UN ARRAY

La soluzione che espongo riguarda il linguaggio C++. Ho notato che tra i programmatori C++ spesso si ha difficoltà nella scrittura degli inzializzatori dei costruttori delle classi che contengono degli array. In queste righe vorrei esporre il modo in cui si scrivono tali inizializzatori, cioè se si ha una classe così definita

Tip fornito dal sig. Alessandro Ghessa

```
    class test1
    {   public:
            // definizioni dei metodi ...
         private:
            int     t[4]; };
```

è possibile inizializzare l'attributo della classe nella sezione degli inizializzatori del costruttore della classe, nel seguente modo

```
    test1() : t((int[4]){1,2,3,4}) {/* codice del costruttore */}
```

anche i singoli elementi dell'array possono essere inizializzati, ad esempio

```
    test1(int a, int b, int c, int d) : t((int[4]){a,b,c,d}) {
                                        /* codice del  costruttore */}
```

stesso discorso se invece dell'array avessimo avuto una struttura, cioè

```
struct teststruct
    {       int     i;
            char    c; };
    class test2
    {  public:
            // definizioni dei metodi ...
    private:
            teststruct      ts;
     };
```

si può inizializzare l'attributo nel costruttore al seguente modo

```
    test2() : ts((teststruct){1,' '}) {}
```

Infine un esempio completo

```
class testclass
{
 public:
 testclass(int a, int b) : i(a), ai((int[3]){b,b,b}) {}
 // altri metodi della classe ...
 private:
   int  i;
 int  ai[3]; };
class test3
{ public:
   test3() : tc((testclass[2]){testclass(1,2),testclass(3,4)}) {}
   test3(int a) : tc((testclass[2]){testclass(a,a),testclass(a,a)}) {}
   test3(int a, int b) : tc((testclass[2]){
                           estclass(a,b),testclass(b,a)}) {}
   // altri metodi della classe ...
 private:
   testclass tc[2]; };
```

```
End With
' Copia l'immagine nella clipboard.
Clipboard.Clear
Clipboard.SetData Picture1.Picture, vbCFBitmap
' L'immagine passa in Word.
WordServer.Selection.Paste
' Le due righe seguenti, chiudono l'applicazione Word
WordServer.Quit True
Set WordServer = Nothing
Screen.MousePointer = vbDefault
Command1.Enabled = True
End Sub
```

C#

CREARE UN DATASET

La seguente funzione crea un *DataSet*, apre una connessione e invia un comando.

```
public System.Data.DataSet dsReturnedDataSetFrom(
                                    System.Data.SqlClient.SqlConnection
sqlOpenConn, System.Data.SqlClient.SqlCommand TheSQLCommand,
                        string TheQueriedTable, string TheDatasetName)
{ TheSQLCommand.Connection = sqlOpenConn;
  System.Data.SqlClient.SqlDataAdapter NewDA =
                            new System.Data.SqlClient.SqlDataAdapter
  (TheSQLCommand);
  DataSet NewDS = new DataSet();
  TheDatasetName.Trim();
  NewDS.DataSetName=TheDatasetName;
  NewDA.Fill(NewDS, TheQueriedTable);
  NewDA.Dispose();
  return NewDS; }
```

COME LIMITARE L'APPLICAZIONE AD UNA SINGOLA ISTANZA

Il codice che segue impedisce che gli utenti possano lanciare più istanze di una stessa applicazione.

```
static void Main()
{
  Process ThisProcess = Process.GetCurrentProcess();
  Process [] AllProcesses = Process.GetProcessesByName(
                                        ThisProcess.ProcessName);
  if (AllProcesses.Length > 1)
  {
    MessageBox.Show(ThisProcess.ProcessName + " is already
       running", ThisProcess.ProcessName, MessageBoxButtons.OK,
                                             MessageBoxIcon.Error);
  }
  else
  {
     Application.Run(new MainForm());
   }
}
```

REGEX PER INIZIALIZZARE UN ARRAY DI STRINGHE

Capita spesso di dover inizializzare un lungo array. La notazione classica, oltre che noiosa, può essere foriera di errori. Il metodo che proponiamo sfrutta la forza delle Regular Expression per inizializzare più rapidamente gli array:

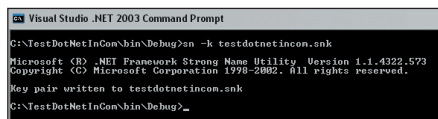
```
String[] s = Regex.Split("Ogni parola di questa stringa sarà un
                                           elemento diverso", " ");
```

Il metodo Split ritorna un array di stringhe contenente gli elementi individuati separando la stringa di partenza, utilizzando come parametri separatori quelli indicati nella seconda stringa. In questo caso, lo spazio.
L'esecuzione di questa farà in modo che la stringa s sarà così costituita:

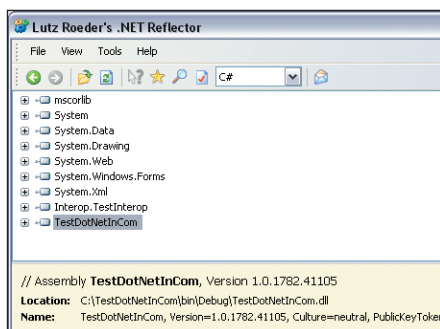
```
s(0) = "Ogni"
s(1) = "parola"
s(2) = "di"
s(3) = "questa"
s(4) = "stringa"
s(5) = "sarà"
s(6) = "un"
s(7) = "elemento"
s(8) = "diverso"
```

Utilizzare componenti .NET all'interno di progetti COM

Abbiamo visto come sia possibile utilizzare componenti COM in applicazioni .NET. È altrettanto semplice scrivere componenti .NET e utilizzarli in ambienti COM. Possiamo così sfruttare la potenza di .NET in un'applicazione VB scritta già da tempo, integrando alcune funzionalità che ci vengono dal framework nelle applicazioni un po' datate. Sono poche le caratteristiche .NET che non possono essere riutilizzate, come i metodi statici, i campi costanti e i costruttori parametrizzati. Un programma COM sfrutta i servizi di un componente .NET attraverso le sue interfacce, e tali interfacce possono essere utilizzate sia con il *late binding*, ovvero chiamando a runtime l'interfaccia voluta, sia con l'early binding, cioè avendo sin dal design time tutte le informazioni necessarie. Per far questo dovremo produrre un file che descriverà le interfacce supportate dal nostro assembly. Tale file è chiamato *type library*: utilizzeremo degli strumenti distribuiti con il framework .NET per produrre la type library, la registreremo e la utilizzeremo per avere l'ausilio dell'Intellisense in un progetto Visual Basic 6. Per prima cosa dovremo scrivere una dll .NET per fare i nostri test.

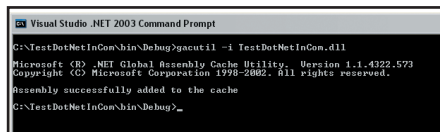
Marco poponi

‹1› IL COMPONENTE .NET

```
using System;
using System.IO;
using System.Runtime.InteropServices ;
namespace TestDotNetInCom
{ public interface ITester
  {string  DammillNomeDelFile(string pathCompleto);}
  [ClassInterface(ClassInterfaceType.None)]
  public class Tester:ITester
  { public Tester(){}
    public string  DammillNomeDelFile(string
                                        pathCompleto)
    { return Path.GetFileName(pathCompleto); } }
}
```

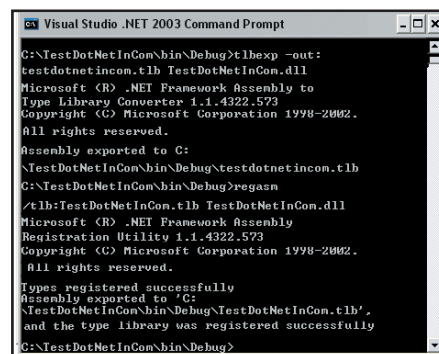
Nel codice proposto vediamo che vengono create una interfaccia, *ITester* e una classe, *Tester*, che implementa l'interfaccia. La classe è decorata con un attributo, *[ClassInterface(ClassInterfaceType.None)]* che regola la creazione della type library. L'uso dell'interfaccia non è obbligatorio, ma permette di scrivere codice client più disaccoppiato dalla implementazione. Consultare MSDN per vedere gli altri valori dell'attributo *ClassInterface*.

‹2› CREIAMO UNO STRONG NAME PER L'ASSEMBLY

```
Visual Studio .NET 2003 Command Prompt
C:\TestDotNetInCom\bin\Debug>sn -k testdotnetincom.snk

Microsoft (R) .NET Framework Strong Name Utility  Version 1.1.4322.573
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Key pair written to testdotnetincom.snk

C:\TestDotNetInCom\bin\Debug>_
```

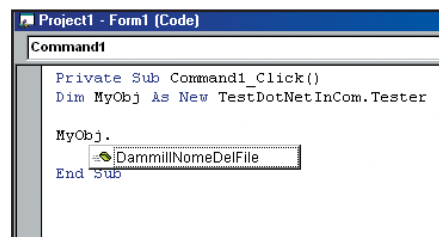
Perché l'assembly possa essere istanziato, deve poter essere trovato dal framework. Per rendere questo possibile registreremo l'assembly nella *GAC*, la *Global Assembly Cache*. A tale scopo dovremo generare uno *strong name* per l'assembly, ovvero un nome che possa far individuare univocamente l'assemby. Dobbiamo quindi creare una chiave, aprendo il prompt dei comandi, ed eseguendo il programma *sn.exe* con il comando riportato in figura.

‹3› COMPILAZIONE

Abbiamo creato un file *.snk*, che dovremo riportare nelle ultime righe del file *AsseblyInfo.cs* del progetto C#. Il path da riportare è relativo alla directory *obj*. Se creassimo il file *testdotnetincom.snk* nella directory di debug, dovremmo scrivere:

```
[assembly: AssemblyKeyFile(
              @"..\..\bin\Debug\testdotnetincom.snk")]
```

Possiamo ora compilare il nostro progetto. Abbiamo creato un assembly con un nome crittografico unico, come visibile dal *PubliKeyToken* in figura.

‹4› REGISTRIAMO L'ASSEMBLY NELLA GAC

```
Visual Studio .NET 2003 Command Prompt
C:\TestDotNetInCom\bin\Debug>gacutil -i TestDotNetInCom.dll

Microsoft (R) .NET Global Assembly Cache Utility.  Version 1.1.4322.573
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Assembly successfully added to the cache

C:\TestDotNetInCom\bin\Debug>_
```

Compilato l'assembly con il suo strong name, utilizziamo lo strumento *gacutil.exe* per inserirlo nella GAC. Il componente è ora pronto per essere utilizzato da client .NET in early binding, per esempio in uno script VBS attraverso il comando *CreateObject*. Per usare invece l'assembly creato a design time da VB 6 o Visual C++ attraverso l'intellisense, dobbiamo produrre la type library, comeve diamo nel punto 5.

‹5› ESPORTIAMO E REGISTRIAMO LA TYPE LIBRARY

La type library è contenuta in un file *.tlb* che descrive le coclass e le interfacce COM esposte dall'assembly .NET. Dovremo poi registrare tale file *.tlb*, in modo che ne possa essere inserito riferimento nel registry di Windows. La produzione e la registrazione della *.tlb* viene fatta attraverso l'uso dei tool *tlbexp.exe* e *regasm.exe*, come mostrato in figura.

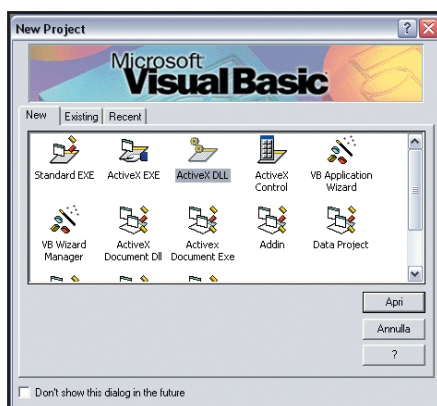
‹6› UTILIZZIAMO L'ASSEMBLY ALL'INTERNO DI VB

Per provare il funzionamento della dll possiamo creare un progetto VB, inserire tra i riferimenti del progetto la *tlb* creata (che apparirà nella lista dei riferimenti COM) e scrivere il codice di utilizzo delle nostre classi.
Nell'immagine si vede come l'intellisense di Visual Basic riconosca la classe e ci mostri il metodo voluto.

Utilizzare componenti COM in .NET con Visual Studio

Ora che siamo passati a .NET dobbiamo buttare tutti i componenti COM che abbiamo scritto? Certo che! È possibile riutilizzarli in maniera semplice e immediata, grazie ai servizi di interoperability. .NET infatti prevede l'utilizzo di classi chiamate *Runtime Callable Wrappers* che operano da proxy fra il mondo COM e il mondo .NET. Una classe wrapper è una classe che ne ingloba un'altra, fornendo all'esterno un'interfaccia differente: nel nostro caso l'architettura COM e .NET. Un oggetto proxy invece viene utilizzato da un client per sfruttare i servizi di un componente remoto o che si rifà ad una differente architettura. Utilizzando Visual Studio la creazione di tali oggetti è del tutto trasparente all'utente. Più in dettaglio: Visual Studio crea una DLL .NET che ingloba e descrive il componente COM originario, in modo tale che il nostro codice .NET non si accorge di chiamare un componente COM. L'IDE crea una DLL il cui nome è "*interop.*" seguito dal nome della DLL originaria. L'IDE crea anche altre DLL, esattamente una per ogni DLL COM da cui l'oggetto inglobato dipende, se ciò è ricavabile dalla sua interfaccia.

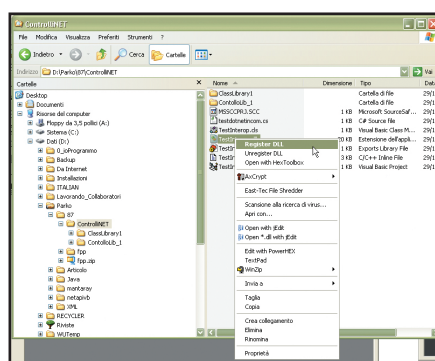
Marco poponi

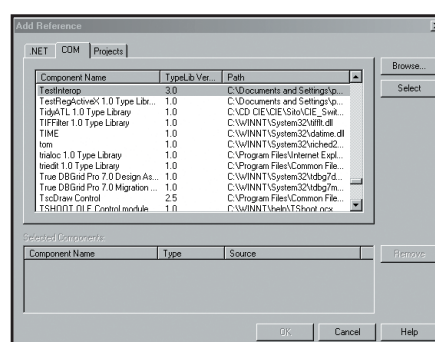
‹1› PER PRIMA COSA PROCURIAMOCI UNA DLL COM

Creiamo in VB un semplicissimo componente COM, che espone una classe, *ClasseDiTest*. Chiameremo il nostro componente *TestInterop.dll* e lo compileremo normalmente.

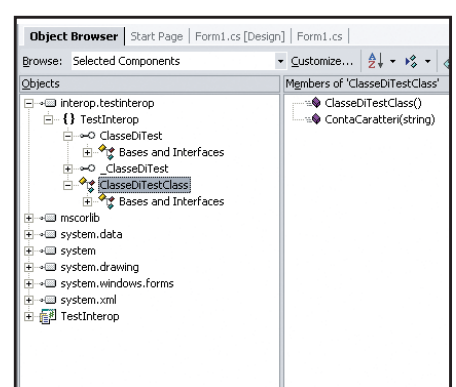
‹2› IL CODICE DEL COMPONENTE VISUAL BASIC

```
Public Function ContaCaratteri(
    ByVal Ingresso As String) As Integer
  ContaCaratteri = Len(Ingresso)
End Function
```

ContaCaratteri è il metodo esposto dal componente COM: prende in ingresso una stringa, e restituisce la lunghezza in caratteri della stringa. Sul fronte Visual Basic, non c'è bisogno di alcun accorgimento particolare per far sì che il componente possa essere riutilizzato in .NET: chi scrive il codice VB può tranquillamente ignorare che la DLL sarà utilizzata in un progetto .NET. A questo punto possiamo compilare la nostra dll COM, assegnandole il *progID* desiderato, nel nostro caso *TestInterop*. Questo *progID*, sarà il namespace del componente una volta importato in .NET, mentre il nome della classe seguito da Class sarà il nome della classe che dovremo istanziare.

‹3› REGISTRIAMO LA DLL

Ricordiamo di registrare il componente COM che abbiamo realizzato. Windows XP consente di effettuare questa operazione semplicemente cliccando con il tasto destro sulla DLL e scendo la voce Register DLL dal menu, come in figura. Ovviamente, è sempre possibile utilizzare il comando dos *regsvr32*.

‹4› LINKIAMO I RIFERIMENTI

Dal menu *Project* di Visual Studio .NET, apriamo la dialog di inserimento riferimenti, tramite la voce di menu *Add Reference…* Scegliamo la linguetta COM, e aggiungiamo il riferimento alla nostra dll COM scegliendo il *progID* dalla lista o sfogliando le cartelle e selezionando la DLL.

‹5› LA CLASSE IMPORTATA

Se andiamo ad esplorare la gerarchia delle classi con l'Object Browser di Visual Studio .NET vediamo come la classe sia stata importata e sia vista come una classe .NET. È stata importata anche un'interfaccia che descrive il comportamento della classe: è l'equivalente della interfaccia COM che Visual Basic ha creato per noi.

‹6› IL CODICE C#

```
TestInterop.ClasseDiTestClass C= new
TestInterop.ClasseDiTestClass ();
string testo= textBox1.Text;
MessageBox.Show (C.ContaCaratteri (
testo).ToString());
```

Una volta importata, la classe può essere utilizzata. Nel nostro esempio viene creato un oggetto di classe *TestInterop.ClasseDiTestClass*, e viene chiamato il metodo *ContaCaratteri* come se fosse una classe .NET. In realtà è proprio così: quella che noi invochiamo è una classe .NET proxy della classe COM vera e propria. Se infatti andiamo a compilare il progetto .NET, nella cartella di produzione troveremo anche una dll .NET chiamata *interop.TestInterop.dll* che si interfaccia con il componente COM. È proprio a questo assembly .NET che il nostro progetto fa riferimento.

Relazioni di tipo molti a molti in MS Access

Il potente strumento delle *relazioni* messo a disposizione dai comuni DBMS (DataBase Managment System), rischia di restare inutilizzato qualora si costruiscano relazioni molti a molti, se non si procede correttamente alla produzione di una nuova tabella di associazione. Ricordo che due tabelle possono essere tra loro in relazione se hanno informazioni in comune. Esistono diversi tipi di relazione. La più ostica da implementare, che è comunque la più generale, è conosciuta come molti a molti. Si ha quando a un elemento (record o tupla) della prima tabella (relazione) corrisponde a molti elementi della seconda e, viceversa, quando a un elemento della seconda corrisponde a molti della prima. È la relazione che sussiste tra professore e alunno, infatti: un insegnante ha più (*molti*) alunni e un alunno a diversi (*molti*) professori. Scopriamo come possa essere facilmente implementata tale situazione in Access, producendo un'opportuna associazione tra i dati mediante lo strumento *relazioni*. Come esempio di riferimento si considera un DB costituito inizialmente da due tabelle: INSEGNANTE e ALUNNO entrambe con identificativi che fungono da chiave, gli attributi: IDI per INSEGNANTE e IDA per ALUNNO. La classe INSEGNA si farà carico di rappresentare la relawione molti a molti, garantendo il controllo sulla congruità dei dati immagazzinati.

Fabio Grimaldi

‹1› CREAZIONE DELLA TABELLA INSEGNANTE

INSEGNANTE : Tabella

	Nome campo	Tipo dati	
🔑	IDI	Contatore	Identificativo insegnante
	Nominativo	Testo	Cognome e nome dell'insegnante
▶	Materia	Testo	Materia insegnata

Generale | Ricerca

Dimensione campo	20
Formato	

Dal raccoglitore di strumenti selezioniamo tabelle. Scegliamo l'opzione *crea in modalità struttura* e definiamo tre attributi che identificano un professore. L'identificativo *IDI* viene marcato come chiave: è buona norma assegnarlo al tipo contatore (in fase di inserimento viene incrementato automaticamente dal sistema). Salviamo con il nome di insegnante e ripetiamo la stessa procedura per l'alunno.

‹2› INSERIMENTO DATI NELLA TABELLA ALUNNO

ALUNNO : Tabella

	IDA	Nome	Classe
	11	Baffa Ester	3B
	9	Capraro Susan	3B
	8	Formica Christian	3B
	2	Gallo Marco	3B
	10	Irianni Pompea	3B
	7	Lumaca Andrea	3B
	6	Mercurio Giada	3B
	12	Morelli Fiorella	3B
	5	Morrone Natale	3B
	3	Pasqua Michele	3B
	4	Reale Rosy	3B
▶	1	Rossi Pierpaolo	3B
*	ontatore)		

Andiamo a popolare entrambe le tabelle create in modalità struttura: *INSEGNANTE* e *ALUNNO*. Cliccando sul nome della tabella *ALUNNO*, apparirà la griglia con i tre attributi come intestazioni. Possiamo così inserire i dati degli alunni. L'identificativo *IDA* iene aggiornato automaticamente. Si ripete lo stesso procedimento per l'insegnante.

‹3› CREAZIONE DELLA TABELLA DI ASSOCIAZIONE: INSEGNA

Tabella1 : Tabella

	Nome campo	Tipo dati	
🔑	IDI	Numerico	Identificativo ansegnante
🔑	IDA	Numerico	Identificativo alunno
▶	Oresett	Numerico	Numero di ore settimanali insegnate

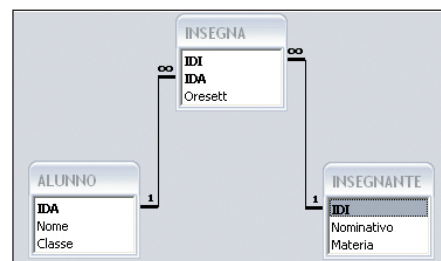
Generale | Ricerca

Dimensione campo	Intero
Formato	
Posizioni decimali	Automatiche
Maschera di input	

Nel caso descritto di relazione molti a molti è necessario creare un'altra tabella. La nuova dovrà contenere le chiavi delle due tabelle da relazionare (IDI e IDA) ed eventuali attributi dell'associazione, nel caso specifico *Oresett*. Per impostare come unica chiave l'unione delle due esterne selezionare entrambe le righe e cliccare sul simbolo di chiave. Chiudendo si salva dando nome *INSEGNA*.

‹4› APPLICAZIONE DELL'INTEGRITÀ REFERENZIALE ALLA RELAZIONE PRODOTTA

Dal menu *strumenti* si seleziona *relazioni*. Nella finestra che appare si scelgono le tre tabelle presenti nel database. Con delle semplici operazioni di drag&drop, evidenziamo e linkiamo gli attributi da mettere in relazione (*IDI* per *INSEGNANTE* e *INSEGNA* e *IDA* per *ALUNNO* e *INSEGNA*). Si applica, come mostrato in figura, l'integrità referenziale marcando tutti i segni di spunta.

‹5› ABBIAMO COSÌ OTTENUTO LA STRUTTURA RELAZIONALE MOLTI A MOLTI

La struttura relazionale finale che si ottiene mostra come la tabella *INSEGNA* funga da elemento associativo. Ogni elemento di tale tabella indicherà il professore (identificativo), l'alunno (identificativo) e le ore settimanali che quel determinato professore ha con il rispettivo alunno. Ecco la relazione molti a molti tra *ALUNNO* e *INSEGNANTE*.

‹6› L'INSERIMENTO DEI DATI È PROTETTO !

INSEGNA : Tabella

	IDI	IDA	Oresett
	1	2	3
	1	3	3
	1	5	3
	2	1	1
	2	10	1
	5	8	4
	5	9	4
	5	10	4
	5	11	4
	5	12	4
✎	5	13	4
*	0	0	0

Microsoft Access

Impossibile aggiungere o modificare il record. Nella tabella "ALUNNO" è necessario un record correlato.

OK | ?

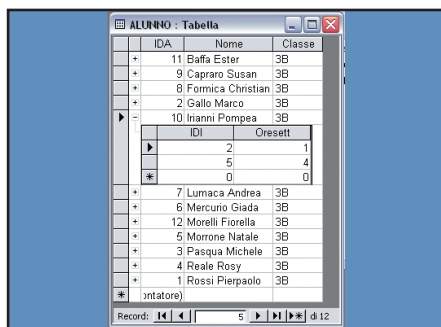
Nell'inserire i dati bisogna specificare gli identificativi dell'insegnante e dell'alunno che sono chiavi delle rispettive tabelle di appartenenza. Così si elimina il problema della ridondanza.

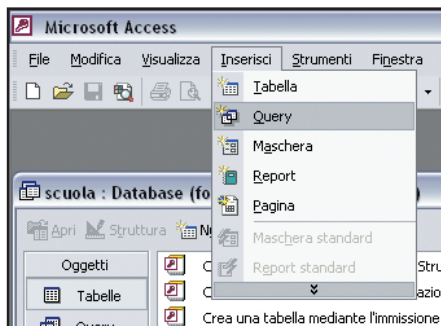
Nel caso si inserisca un dato non corrispondente (esempio, alunno con IDA 13 che non esiste) si genera in automatico un errore essendo le tabelle in relazione.

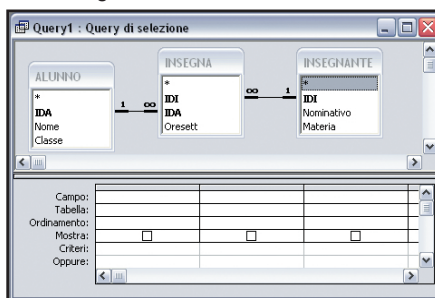
Query in MS Access per la simulazione di operazioni di join

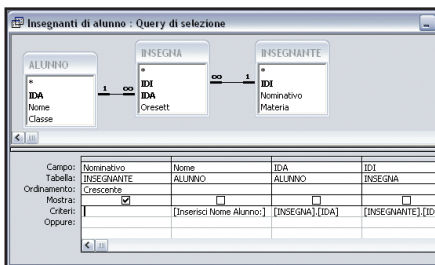
Aprire delle finestre informative (viste di dati) all'interno di un DB è un operazione di fondamentale importanza. MS Access a tale scopo mette a disposizione dell'utente le query, con le quali è possibile prelevare porzioni di informazioni di nostro interesse attingendo a quella che spesso è un enorme mole di dati, ovvero il DB nella sua globalità . Ma se il compito di interrogare un DB risulta semplice nel caso in cui il DB coincida con una sola unità informativa, ossia un'unica tabella di dati, la cosa si complica se vi sono più tabelle che sono tra loro in relazione. Il compito più ostico si presenta qualora sia concretizzata una relazione di tipo molti a molti. Tale relazione, a partire da due tabelle di dati, ne produce una terza come associazione delle due che contiene le chiavi delle sopraccitate e gli eventuali attributi della relazione. In tal caso, al fine di interrogare il DB è necessario simulare le operazioni di join con le quali si possono unificare più tabelle in base alla condizione di eguaglianza tra due attributi comuni. Nell'esempio proposto il DB SCUOLA è costituito dalle due tabelle di dati INSEGNANTE e ALUNNO e dalle tabella INSEGNA, che con la presenza delle chiavi delle due relazioni iniziali, consente di attuare l'associazione. La query deve individuare i professori di un generico alunno inserito da input.

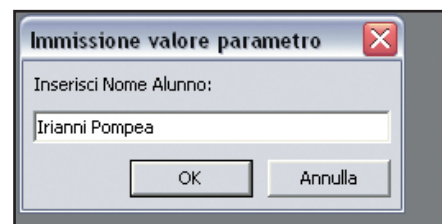
Fabio Grimaldi

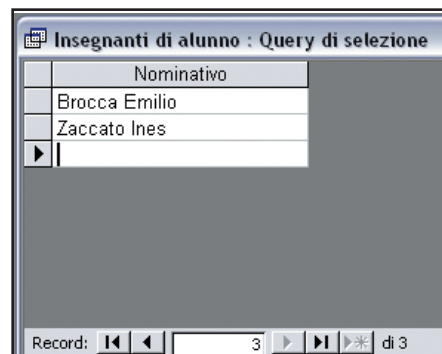
‹1› APRIAMO IL DB SCUOLA

Dopo aver avviato Access, e aver aperto il *DB SCUOLA*, diamo un'occhiata agli elementi a disposizione. Dal raccoglitore di strumenti notiamo la presenza di tre tabelle. Apriamo *ALUNNO* e clicchiamo sul segno più (+) accanto ad un record qualsiasi: possiamo leggere una relazione con una nuova tabella *INSEGNA* che indica i suoi docenti (identificativi).

‹2› ATTIVIAMO UNA NUOVA QUERY

Dal menu *inserisci* scegliamo la voce *query* (risultato analogo si ottiene selezionando lo strumento *query* dall'apposito raccoglitore che appare quando si apre il DB). Tra le modalità proposte, scegliamo *"struttura"* perché più flessibile. In generale, quando è possibile, è buona norma evitare le creazioni guidate, sono più rapide ma ci obbligano a seguire percorsi prestabiliti.

‹3› DEFINIAMO LA STRUTTURA DELLA QUERY

Nella modalità struttura della query il primo passo è quello di individuare le tabelle che saranno sottoposte ad esame. Nel caso specifico sono tutte. Notiamo che, dopo averle aggiunte, esse conservano le caratteristiche di relazionalità. Si tratta adesso di impostare la query. Si vogliono conoscere i professori di un generico alunno proposto da input.

‹4› DEFINIAMO I CRITERI DELLA QUERY

Cliccando sugli attributi si delinea la query. Si lascia il segno di spunta solo sull'attributo che si vuole visualizzare: il nominativo del professore. Aggiungendo come criterio di eguaglianza all'attributo IDA di *ALUNNO* il valore *INSEGNA.IDA* si realizza la prima join.
Analogamente si procede con la seconda. Infine, il criterio del nome dell'alunno mostrato consente l'input.

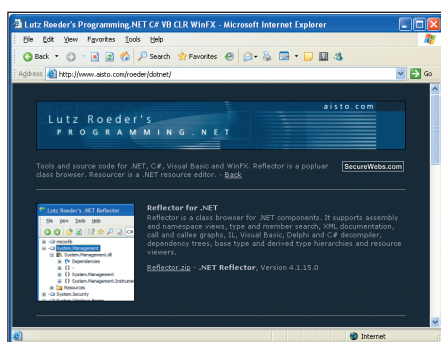
‹5› L'ESECUZIONE DELLA QUERY RICHIEDE UN VALORE DI INPUT

Salvare la query con il nome di *"Insegnanti di alunno"* fa sì che, quando viene richiamata, si attiva il criterio che avevamo posto al passo precedente sul nome alunno. Viene richiesto il nome dell'alunno di cui si vogliono conoscere i professori. Scegliamo uno dei presenti, ad esempio *"Irianni Pompea"*, e clicchiamo su *OK*.

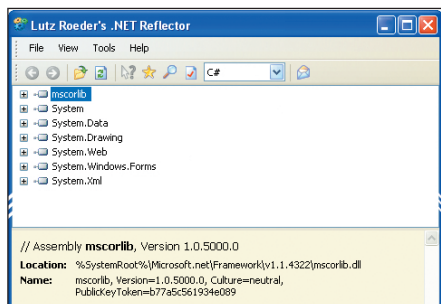
‹6› VISUALIZZAZIONE DEL RISULTATO OTTENUTO

Non resta che esaminare il risultato ottenuto. In automatico apparirà una nuova tabella (in teoria conosciuta come relazione virtuale, poiché ottenuta da relazioni di base) costituita dal solo campo nominativo, così come l'avevamo pensata e sviluppata, che indica i nomi dei professori dell'alunna "Irianni Pompea". Obiettivo raggiunto!!

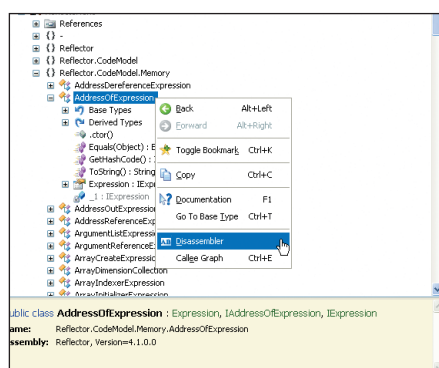
Alla scoperta degli assembly .NET con Reflector

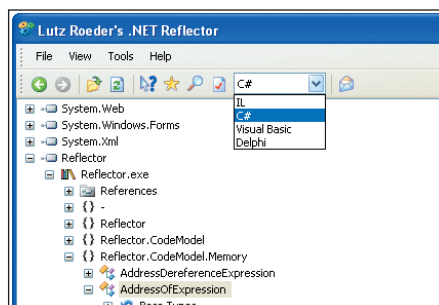
Quando si ha a che fare con lo sviluppo di applicazioni .NET, spesso si presenta la necessità di conoscere il contenuto di un assembly che si sta utilizzando, capire il funzionamento di qualche metodo o classe particolare (magari contenuti in uno dei tanti namespace messi a disposizione dal framework .NET stesso), oppure semplicemente si desidera sapere, a scopo didattico, come sono implementate specifiche funzionalità del nostro software freeware preferito. Per queste e molte altre "problematiche" simili, viene in nostro aiuto Reflector, un tool potente e gratuito scritto da Lutz Roeder che consente la navigazione e la ricerca all'interno dei metadatai, del codice IL, delle risorse e della documentazione XML memorizzate negli assembly .NET. Si tratta di un tool che dovrebbe essere presente nella cassetta degli attrezzi di ogni sviluppatore della piattaforma MS e che, una volta scoperte le innumerevoli funzionalità, diventerà certamente un tool di cui non si può fare a meno. Vediamo passo per passo come utilizzare le sue caratteristiche principali.

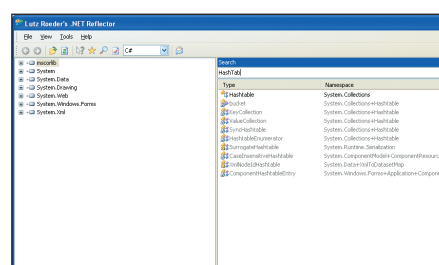
Fabrizio Bernabei

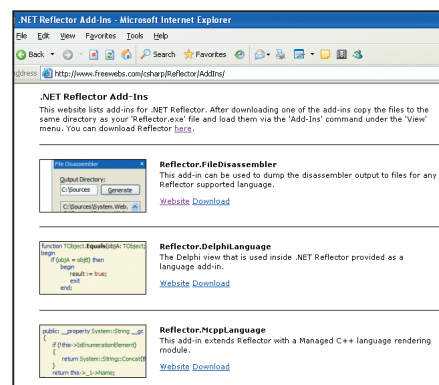
<1> INSTALLAZIONE

L'installazione di Reflector è estremamente semplice in quanto il pacchetto è distribuito sotto forma di file zip contenente l'eseguibile ed un file html che ne spiega brevemente le funzionalità. Dopo aver scaricato la versione più recente all'indirizzo **www.aisto.com/roeder/dotnet/**, non resta che decomprimere il file. Ovviamente, trovate il file di installazione fra i software presenti nel CD di questo mese. Potete dunque evitare di scaricare il file dal Web...

<2> L'INTERFACCIA

Al primo avvio ci viene mostrata la lista delle versioni del framework presenti sulla macchina. Selezionandone una verranno caricati gli assembly contenenti le classi della libreria .NET, rappresentate con una visualizzazione ad albero dove le radici rappresentano i namespace di sistema e, scendendo in "profondità", si arriva alle foglie che identificano metodi, proprietà, ecc..

<3> ESPLORAZIONE DEGLI ASSEMBLY

Dal menu *File->Apri* è possibile caricare altri assembly da esplorare, per esempio lo stesso eseguibile di Reflector, anch'esso scritto in .NET. Aprendo poi una qualsiasi classe (o interfaccia, enumerazioni o qualsiasi altro tipo .NET), vengono mostrate tutte le informazioni del caso come tipi da cui deriva, interfacce implementate o accessibilità dei metodi.

<4> DISASSEMBLAGGIO DEL CODICE

Premendo il tasto destro su di un qualsiasi elemento, si accede a tutte le funzioni avanzate di Reflector, tra le quali risulta molto utile a scopo didattico quella di disassemblaggio. Utilizzandola ci verrà mostrata nell'area destra il codice (disponibili C#, VB.NET, Delphi.NET o IL) dell'implementazione dell'oggetto selezionato.

<5> ALTRE FUNZIONALITÀ

Fra le altre funzioni disponibili, risultano molto utili quelle per visualizzare tutti i riferimenti ad un oggetto (sempre accessibili dal menu contestuale), necessarie ad esempio per vedere tutte le occorrenze di un determinato metodo, o tutti i metodi che restituiscono o accettano come parametro un determinato tipo, oltre all'utilissima funzione di ricerca.

<6> POTENZIAMENTO DI REFLECTOR

Come se non bastasse, Reflector offre agli sviluppatori .NET la possibilità di creare *add-in* (una lista molto interessante si può trovare all'indirizzo **www.freewebs.com/csharp/Reflector/AddIns/**) per estendere le funzionalità del software. Meritano una menzione *Reflector.Diff* (differenze tra due assembly), *Reflector.VisualStudio* (integrazione nell'IDE Microsoft) e *Reflector.Graph* (visualizzazione grafica delle dipendenze).

ADO.NET: Uso di DataAdapter dinamici per l'accesso ai dati

Il framework .NET ha cambiato radicalmente il modo di lavorare ed accedere ai dati. L'introduzione di classi come il *Dataset* ed il *DataAdapter* ha permesso agli sviluppatori di realizzare soluzioni potenti e flessibili, e il supporto degli IDE (primo fra tutti MS Visual Studio) con strumenti visuali e wizard per la generazione di classi di accesso ai dati, ha reso lo sviluppo di applicazioni sempre più facile e veloce, permettendo, ad esempio, di generare per mezzo di tool visuali tutte le classi che gestiscono dati senza scrivere una riga di codice. Se da un lato questi strumenti velocizzano notevolmente lo sviluppo, dall'altro tendono a rendere il codice difficilmente manutenibile al crescere delle dimensioni. Basti pensare a cosa succede se le classi *DataAdapter* sono generate staticamente: una modifica al DB significa rivederne gran parte per adeguarle alle modifiche. Una possibile soluzione consiste nello scrivere poche righe di codice per rendere dinamica la gestione delle classi di accesso ai dati, ovvero generare dinamicamente i *DataAdapter* quando servono, in modo da non vincolarsi alla struttura del DB.

Fabrizio Bernabei

‹1› L'APPLICAZIONE IN ESECUZIONE

L'applicazione di esempio, una volta terminata, sarà in grado di accedere ai dati contenuti in un file MDB (utilizzando Dataset tipizzati e non) e permetterà la modifica/inserimento/cancellazione dei record. Da notare che per la visualizzazione delle informazioni è stato utilizzato un controllo *DataGrid* che gestisce in modo automatico anche l'editing dei dati contenuti.

‹2› CREAZIONE DINAMICA DEL DATAADAPTER

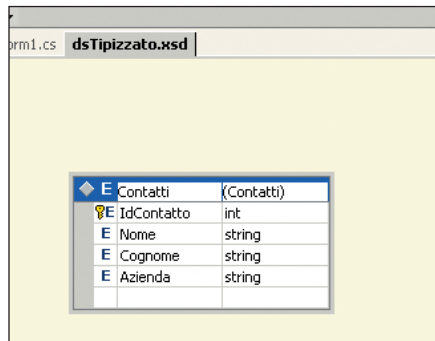
```
using System.Data.OleDb;
...
public OleDbDataAdapter CreaDataAdapter(
                              string NomeTabella )
{
   string sql = "select * from " + NomeTabella;
   OleDbDataAdapter da = new OleDbDataAdapter(
                         sql, oleDbConnection);
   da.TableMappings.Add("Table", NomeTabella);
   return da;
}
```

Sono sufficienti due semplici passi per creare un *DataAdapter* "al volo". La creazione dell'istanza della classe, fornendo come parametri un comando SQL da eseguire (nel nostro caso l'istruzione che ci permette di leggere una tabella completa), e l'aggiunta nella collezione dei *TableMappings* del nome da utilizzare per i dati restituiti (al posto del default '*Table*').

‹3› CARICAMENTO DEI DATI NEL DATASET

```
DataSet ds = new DataSet();
try
{  CreaDataAdapter("Contatti").Fill( ds );
   dataGrid1.DataSource = ds.Tables[0];
}
catch(OleDbException exc)
{
   MessageBox.Show("Errore: "+exc.Message);
}
```

Per la lettura dei dati della tabella *Contatti* basterà costruire il *DataAdpter*. Con questo *DataAdpter* potremo un *DataSet* non tipizzato per mezzo della chiamata del metodo *Fill*, ed infine visualizzeremo i dati caricati. Da notare che il *DataAdapter* si occupa automaticamente di aprire e chiudere la connessione al DB, quando necessario.

‹4› USO DI DATASTE TIPIZZATI

Lo stesso procedimento può essere utilizzato anche per popolare un *Dataset* tipizzato. Differenze rispetto al caso precedente sono visibili solo a livello di stesura del codice, e in questa sede non ci preoccuperemo di "dibattere" su quale sia il modo migliore. Ci basterà sapere che questo approccio per la costruzione dei *DataAdapter* è utilizzabile in entrambi i casi.

‹5› ALTRE FUNZIONALITÀ DEL DATAADAPTER

```
..
 OleDbCommandBuilder cb =
                    new OleDbCommandBuilder(da);
   oleDbConnection.Open();
   a.InsertCommand = cb.GetInsertCommand();
   a.UpdateCommand = cb.GetUpdateCommand();
 da.DeleteCommand = cb.GetDeleteCommand();
 oleDbConnection.Close();
 return da;
}
```

Il metodo *CreaDataAdapter* permette solamente la lettura dei dati. Per supportare anche le operazioni di modifica dovremo aggiungere, per mezzo della classe *CommandBuilder*, anche i comandi di modifica e cancellazione al *DataAdapter* da restituire. In questo caso, la classe *CommandBuilder* non gestisce automaticamente la connessione.

‹6› SALVATAGGIO DEI DATI

```
if( dataGrid1.DataSource != null )
{
OleDbDataAdapter da = CreaDataAdapter( "Contatti" );
da.Update(dataGrid1.DataSource as DataTable);
}
```

Non resta altro che scrivere il codice per il salvataggio dei dati modificati, ed il nostro esempio è terminato. Va sottolineato il fatto che l'approccio utilizzato è realizzabile con tutti i Provider ADO.NET, anche se per comodità nel nostro caso abbiamo scelto gli OleDbProvider. Le uniche modifiche da implementare in questo caso riguardano ovviamente i nomi delle classi che cambieranno in base al DB scelto (per esempio si useranno SqlDataAdapter e SqlCommandBuilder per l'utilizzo con Sql Server), mentre i metodi usati restano invariati.

SOFTWARE SUL CD

INTERNET

Amsn 0.94

Un clone di MSN Messenger, completo di sorgenti

La novità più importante di questa versione è un rinnovato sistema per la gestione dei plugin. Con il nuovo sistema, qualunque programmatore può estendere le funzionalità di AMSN.
Due plugin d'esempio simulano, rispettivamente, l'azione */ME* tipica delle chat e il *Nudges*

Directory: /amsn

Azureus 2.2.0.0

Il client BitTorrent multipiattaforma

Se avete provato qualche volta a scaricare una versione di Linux, vi sarete accorti che la maggior parte delle distribuzioni consente di downloadare le iso in una forma chiamata "BitTorrent".
Si tratta di una forma di P2P molto particolare, usato spesso proprio in congiunzione al download di file di grandi dimensioni, come ad esempio le distribuzioni Linux.

File: Azureus_2.2.0.0_Win32.setup.exe

Eggdrop 1.6.17

Gestisci un canale irc con il tuo bot programmabile

Se siete dei frequentatori abituali delle Chat IRC sapete già cosa è un bot. Quello che è più interessante è che la maggior parte dei bot che mantengono in piedi i vari canali Irc sono dei software altamente programmabili realizzati con Eggdrop.
Il linguaggio di programmazione utilizzato per la programmazione di un Eggdrop è TCL. Gli Eggdrop funzionano normalmente in ambiente Unix. Ne esiste un corrispondente per Windows: Windrop.

Directory: /eggdrop

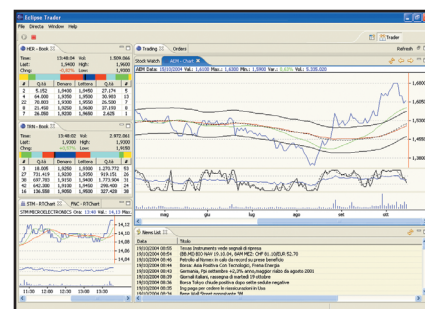
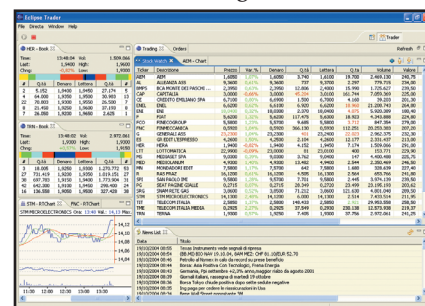
WordPress 1.2.1

Un ottimo tool per la creazione di blog

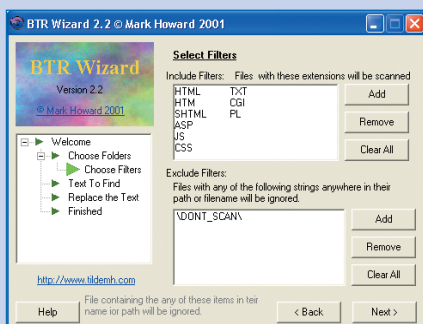
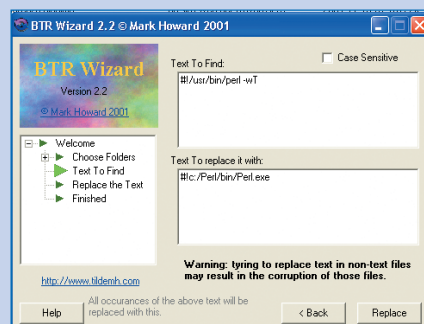
La nuova era di Internet è nata! I Blog sono la nuova frontiera. Piccoli sistemi di Content Management che consentono a chiunque di trasferire in modo rapido dei contenuti sul Web. WordPress è uno dei sistemi piu amati dai Blogger. Scritto in PHP è dotato di moltissimi moduli preconfezionati. Ovviamente è possibile creare dei moduli propri.

Directory \wordpress

Elicpse Trader 0.90

Un plugin per Eclipse per costruire applicazioni di stock trading

File: eclipsetrader-0.7.0-win32.zip

SOSTITUIRE UNA STRINGA DI TESTO IN CENTINAIA DI FILE IN UN SOL COLPO CON BTR WIZARD

1 Selezioniamo la directory in cui sono contenuti i file su cui effettuare la ricerca

2 Stabiliamo se la ricerca deve essere compiuta su tutti i file o limitarsi ad alcune estensioni

3 Diciamo a BTR Wizard quale stringa deve essere ricercata e con cosa deve essere sostituita

File: btrwiz_22.exe

Quick 'n Easy FTP Server 2.4.2

Un server FTP da installare con un clic!

Completamente gratuito, questo server FTP si presenta semplicissimo da installare: un doppio clic per avviare il programma e saremo guidati, attraverso un semplicissimo wizard, alla definizione di tutte le opzioni necessarie al corretto funzionamenti del server. L'interfaccia per la gestione degli account è di una chiarezza esemplare e garantisce anche ai meno esperti un utilizzo corretto ed efficiente del server. Non mancano le principali funzionalità come il supporto per le directory virtuali, l'abilitazione ed il ban per specifici IP e la definizione della larghezza di banda disponibile per ogni utente.

ftpserver2.zip

PHPBB 2.0.10

Un forum preconfezionato in PHP, pronto per essere usato

PHPBB è diventato ormai uno standard per la creazione e gestione di Forum su Internet.
Si tratta di un applicativo scritto in PHP, ormai maturo, dal numero elevatissimo di funzionalità, facilmente estendibile e personalizzabile.

File: phpBB-2.0.10.zip

Deep Log Analyzer 1.51

Per raccogliere e analizzare statistiche sul Web

La visualizzazione per via grafica delle statistiche è il punto di forza di Log Analyzer: il numero e la complessità delle diverse analisi possibili consentono di avere in ogni momento il quadro aggiornato dell'accesso ai nostri siti. I raggruppamenti che abbiamo a disposizione vanno a definire un ampio campionario di possibili report mentre la grafica evoluta consente una efficace interattività con l'utilizzatore.
Versione di prova valida venticinque giorni.

dlatrial.exe

STRUMENTI

BTR Wizard 2.2

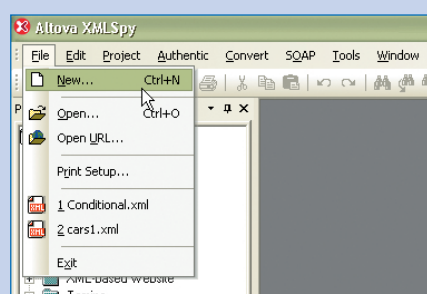
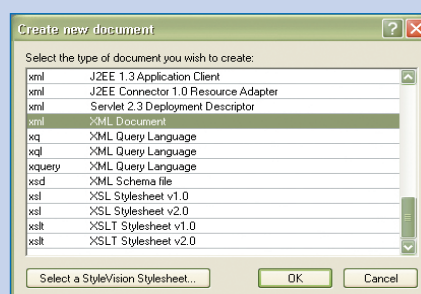
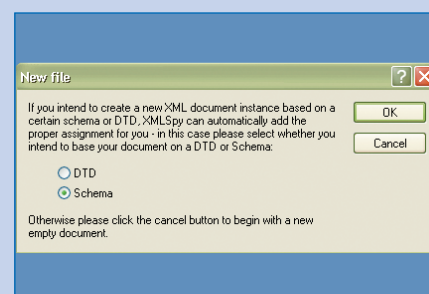
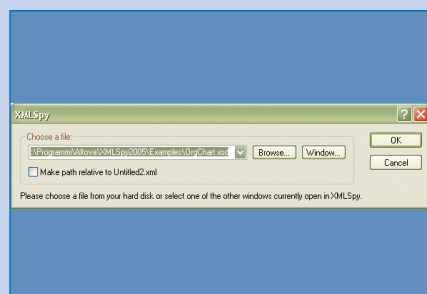
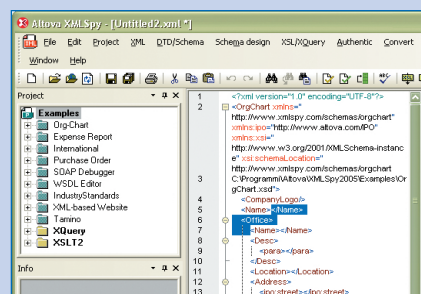
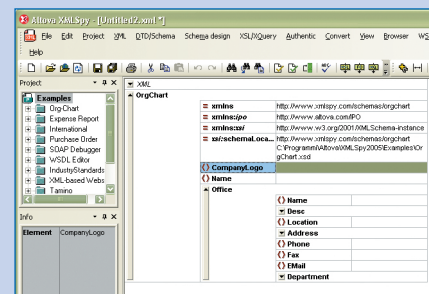
Si tratta di un utilissimo Text Replacer

Noi stessi l'abbiamo utilizzato nell'installazione di BugZilla per sostituire una stringa di testo da ricercare in un centinaio di file di testo con un'altra. Prima di tutto c'è da dire che è velocis-

XMLSPY ENTERPRISE EDITION 2005

Uno dei più completi IDE per lo sviluppo di progetti XML presenti sul mercato

XML Spy 2005, che qui presentiamo nella versione Enterprise, consente la manipolazione di documenti DTD, XML Schema, file XML e fogli XSLT. XMLSPY Enterprise Edition è tra i migliori tool disponibili: agevola la costruzione di applicazioni XML-based, siti Web, Web Services e tutto quanto ruota attorno all'XML, la stella incontrastata dell'attuale panorama della programmazione. È ovviamente possibile validare documenti XML sulla base di DTD ed è possibile trasformare i documenti utilizzando regole XSL, il tutto attraverso un apposito editor DTD, un editor grafico per XML Schema ed un processore XSLT. Al primo avvio, dovrete cliccare sul pulsante "Request a FREE evaluation Key": fornendo il proprio nominativo ed un indirizzo mail, vi sarà in pochi istanti inviata una chiave di attivazione. Da questo momento, avete trenta giorni di tempo per provare la versione Enterprise di XmlSpy; allo scadere di questo periodo, potrete utilizzare XmlSpy in versione Home Edition, limitata in alcune funzionalità ma senza limiti di tempo.

File: XMLSpyEnt2005.exe

1 Per creare un nuovo documento XML, è necessario selezionare la voce New dal menu File.

2 Dal pop up che appare, tra le numerose tipologie disponibili, indichiamo "XML Document"

3 A questo punto ci viene chiesto se impostare il documento su uno Schema o su un DTD

4 Possiamo scegliere un Schema presente su disco o indicarne uno aperto in XML Spy

5 Ecco la classica vista ad albero attraverso cui interagire con il nostro documento

6 Cliccando sul tab "Grid" possiamo accedere ad una vista particolarmente efficace per il data entry

simo. Poi c'è da aggiungere che è molto semplice da usare. Perciò uno strumentino molto utile, che senza troppe complicazioni ci risolve un problema piuttosto frequente.

File: btrwiz-22.exe

Bugzilla 2.18

Gestisci i bug scovati nei tuoi programmi

Quello che capita sempre, dopo avere rilasciato un software, è che gli utenti scovino una serie di Bug inaspettati.
Bugzilla è una comoda interfaccia via web per la gestione di Bug e segnalazioni da parte degli utenti.
È tra l'altro lo stesso identico software utilizzato per tracciare i Bug del Kernel di Linux o quelli di PHP. È scritto in Perl e chi lo usa garantisce che con questo sistema ha raddoppiato la capacità di mettere a punto le nuove versioni del software.

File: bugzilla-2.18rc3.tar.gz

Filezilla 2.2.9

Un client FTP freeware e completo di sorgenti

Molto interessante questo client FTP. Non solo perché può essere utilizzato in modo ottimale per gestire le proprie connessioni FTP durante la giornata di lavoro, ma anche perché la presenza dei sorgenti consente di studiare accuratamente come questo genere di applicazioni può essere implementato. Inoltre, il software può essere esteso secondo le proprie esigenze.

Directory: \filezilla

LE NOVITÀ DI MySQL 4.1

La nuova versione di MySQL soddisfa molte delle richieste che i milioni di utilizzatori nel mondo hanno sempre rivolto a quelli di MySQL AB

Se è vero che un grande passo avanti era già stato fatto con la versione 4.0, alla versione 4.1 sono state aggiunte:

SUPPORTO ALLE SUBQUERY

- È possibile adesso eseguire una *select* su un insieme di dati costruito sulla base di un'altra select, e tutto in un'unica istruzione.

COPIA STRUTTURA TABELLA

- Utilizzando la sintassi
  - *CREATE TABLE **tbl_name2** LIKE **tbl_name1***

  La tabella *tbl_name2* viene riprodotta con la stessa struttura di *tbl_name1.*

SUPPORTO PER OPENGIS

- Questa è decisamente una funzione raffinata, che consente di immagazzinare dati di tipo geografico per ottimizzare l'uso del database MySQL con applicazioni che fanno riferimento a mappe geografiche di qualunque tipo.

MIGLIORATO SUPPORTO AL DEBUGGING

- Sono stati inseriti una serie piuttosto dettagliata di eventuali messaggi d'errore che vengono restituiti in relazione all'esito di un'operazione. Il dettaglio dei messaggi consente un più facile debugging.

UN INSTALLER COMPLETAMENTE RINNOVATO

- Per gli utenti Windows la procedura di Setup è stata completamente rivista. La procedura di installazione include anche un wizard che effettua le configurazioni di base.

File: mysql-4.1.7-win.zip

LIBRERIE

ASP

Visualizzazione dati e WAP

Connessione a un database utilizzando ASP e WAP

[wapdata.zip]

Progress Bar in ASP.NET

Tutto quanto necessita per implementare e personalizzare una barra di avanzamento.

[ASPITALIA1003.ZIP]

PanelSuite

Aggiungi nuove funzionalità come panel bar alla tua applicazione ASP.NET.

[00024145.exe]

Rich Date Picker

Un calendario (ma non solo) pop-up completo per le vostre pagine Web.

[00023950.exe]

JasperReports 0.6.1

Una libreria scritta in Java per creare report personalizzati

Un libreria per generare report in grado di inviare il proprio output allo schermo, alla stampante o addirittura a file in formato PDF, HTML, XLS, CSV e e XML.

Interamente scritta in Java, può facilmente essere utilizzata in un'infinità di applicazioni, per generare contenuti dinamicamente.

File: jasperreports-0.6.1-project.zip

Filezilla Server 0.9.3

Un server FTP affidabile e completo di sorgenti

Se la versione Client di Filezilla rappresenta un'ottima soluzione per la gestione del download FTP, Filezilla Server ne rappresenta il contraltare lato Server.
Anche in questo caso il prodotto è affidabile e la presenza dei sorgenti consente da un lato di studiare la tecnica con cui si può realizzare un server FTP, dall'altro di personalizzarlo per le proprie esigenze.

FileZilla_Server_0_9_3.exe

Hibernate 0.4.0

Sviluppa classi persistenti in Java utilizzando un linguaggio comune

Un framework *"object relational mapping"* si occupa di risolvere in maniera efficace e trasparente i problemi di

coesistenza tra paradigma ad oggetti e paradigma relazionale, proponendo una via che permette di rendere persistente un oggetto salvandolo in maniera automatica su database.
File: nhibernate-0.4.0.0.zip

Ireports 4.0

Un tool di reportistica basato su JasperReports

Se non vi fidavate delle capacità della libreria JasperReports, potete provare questo tool di reportistica basato appunto sulla libreria in questione.
È veramente completo!
Gestisce numerose fonti di dati, incluse XML, ODBC, CSV ed è in grado di creare dei report assolutamente professionali.
Inoltre è completamente FreeWare.
Directory \ireport

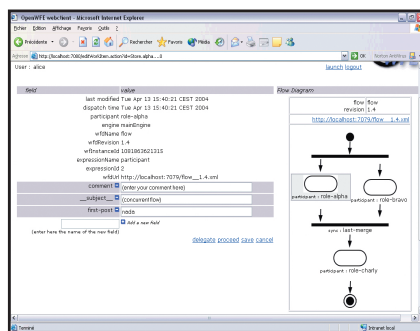
Jaxlib 0.6.3

Una libreria che fornisce una serie di funzionalità per la gestione dell'I/O

Fornisce delle primitive per gestire in modo efficace strutture in memoria e lo stream I/O.
Multipiattaforma, OpenSource e dotata di innumerevoli utility. È senza dubbio un progetto interessante.
File: jaxlib-0.6.3.zip

OpenWFE 1.4.5

Un engine Workflow per Java. Automatizza il processo di lavoro necessario per lo sviluppo

C'è da dire intanto che si tratta di un'applicazione JSP, quindi gira tranquillamente in ambiente Web sia Internet che Intranet.
Un'applicazione veramente completa, capace di gestire gruppi di lavoro coinvolti in progetti anche di grandi dimensioni.

Directory: \openwfe

LIBRERIE

ASP.NET Persistent TreeView

Aggiungere persistenti strutture ad albero alle tue applicazioni ASP.NET.
[00016243.exe]

Estrazione casuale

Una scelta random da un database. Volendo estrarre da una tabella di un database un record, in modo del tutto casuale, non ci resta che affidarci a questo stralcio di codice che implementa proprio tutto ciò.
[randomset.zip]

Currency Server

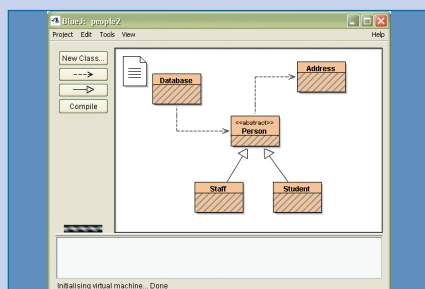
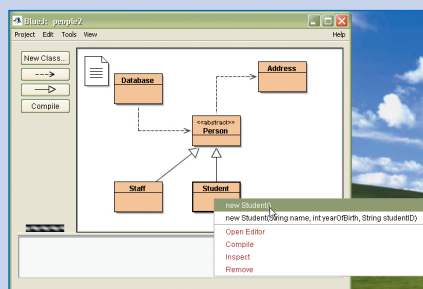
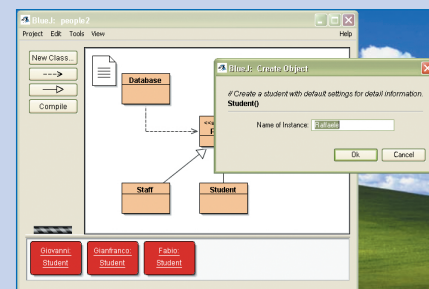
Un completo "Agente di Cambio" sul computer/web.
[00013483.exe]

PinToolbar

Per aggiungere alla tua applicazione una toolbar indipendente.
[00023889.exe]

BLUEJ 2.0.2

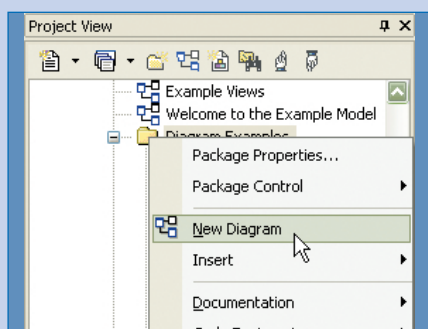
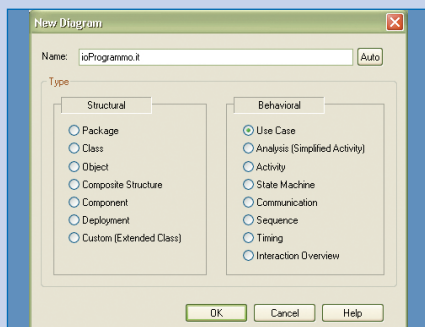
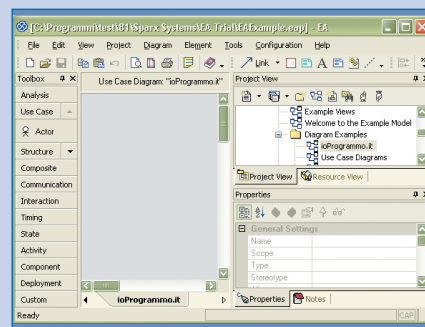
Un'ottima occasione per imparare Java

Un ambiente di sviluppo integrato, creato appositamente per facilitare l'apprendimento di Java e delle tecniche di programmazione Object Oriented. Attraverso una interfaccia grafica particolarmente accattivante e facile da usare, potremo sviluppare applicazioni graficamente per poi lasciare che BlueJ le traduca in Java. BlueJ è un interessante strumento didattico per l'apprendimento della programmazione orientata agli oggetti in Java. BlueJ è un ambiente specificatamente pensato per facilitare l'apprendimento dei concetti fondamentali della programmazione Object Oriented, grazie ad una estrema semplicità d'uso e ad una interfaccia grafica che permette di interagire direttamente con le classi e gli oggetti in modo visuale e al tempo stesso di non perdere di vista il codice Java sottostante. Questo facilita moltissimo chi si impegna a studiare Java come nuovo linguaggio, consentendogli di sperimentare da subito il codice sviluppato, senza dover conoscere gli aspetti più problematici della programmazione in Java. Gratuito
File: bluejsetup-202.exe

1 **BlueJ, oltre all'esecuzione dell'intera applicazione, consente l'interazione diretta con le classi. Le righe oblique presenti sulle classi, indicano che il progetto non è stato ancora compilato. Con la combinazione di tasto *CTRL+K*, effettuiamo la prima compilazione.**

2 **Cliccando con il tasto destro su una delle classi, possiamo ispezionare i metodi costruttori disponibili.
Selezionando il primo, indichiamo il nome di un nuovo studente. Iterando il procedimento, possiamo creare una intera classe!**

3 **Nella finestra in basso, in rosso, sono presenti tutte le istanze degli oggetti creati. È possibile interagire direttamente con tutti gli oggetti: con un clic del mouse, si possono richiamare i metodi esposti e simulare "live" il comportamento di un'applicazione.**

ENTERPRISE ARCHITECT 4.10.739

Per sviluppare e documentare software Object Oriented

Un completo ambiente visuale per sviluppare e mantenere software object oriented. Con il pieno supporto di UML 2.0 e di tutti i diagrammi contemplati dallo standard. Enterprise Architect consente di verificare l'integrità delle dipendenze fra i vari oggetti che compongono i nostri progetti e permette di modellare la gerarchia delle classi, sia staticamente sia dinamicamente.
Con Enterprise Architect è possibile documentare con precisione e rapidamente qualsiasi progetto di sviluppo software, curando tutte le fasi del ciclo di vita del progetto: dalla ideazione alla confezione finale, passando per la fase di test e del controllo sui cambiamenti. Enterprise Architect supporta numerosi tipi di diagrammi UML e, agli utenti più esperti, è lasciata la possibilità di estenderne le capacità con nuovi oggetti.
Enterprise Architect può essere integrato facilmente in team che già usino altri prodotti per la generazione di UML: è infatti possibile importare modelli già esistenti in formato XMI (XML Metadata Interchange).
Insostituibile nelle pratiche di reverse engineering, grazie alla possibilità di riconoscere codice scritto in tutti linguaggi più diffusi: C++, Java, Visual Basic, Delphi, PHP, C# e VB.NET.
Anche chi si occupa di data modelling potrà utilizzare con profitto Enterprise Architect grazie al pieno supporto per SQL e ODBC: a partire da una base di dati già installata e funzionante, sarà possibile risalire alla struttura e a tutti i dettagli degli oggetti che la compongono. Molto efficace anche nelle funzionalità di report che possono essere generato sia in formato RTF sia in HTML. È possibile integrare l'azione di Enterprise Architect con numerosi altri strumenti grazie alle possibilità offerte dall'import/export in formato XML.Versione di valutazione valida trenta giorni.

easetup.exe

1 **Nel *project browser* clicchiamo con il tasto destro sul package in cui vogliamo inserire il diagramma. Selezioniamo *New Diagram* dal *context menu.***

2 **Scegliamo il tipo di diagramma che vogliamo inserire ed indichiamo un nome di identificazione.**

3 **Con un clic su OK, avremo a disposizione un nuovo diagramma su cui cominciare a lavorare.**

LIBRERIE

UIBundle

Per aggiungere alla vostra applicazione ASP.NET un menu di navigazione ed una mappa del sito.

[00024139.exe]

SitemapSuite 1.0

Per disporre della mappa del vostro sito e del suo albero di navigazione.

[00024147.exe]

VISUAL BASIC

List All

Un listview per avere l'elenco dei file di una cartella.

[LISTALL1086237202002.ZIP]

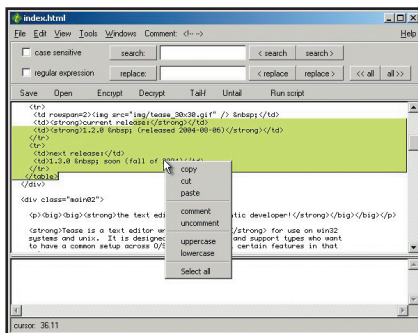
Winsock File Transfer

Un esempio di Winsock File Transfer in VB.NET con clock a 1MB/sec

[AWINSOCK608703102002.ZIP]

Tease 1.2.1

Un editor di testo per sviluppatori, scritto in TCL

Tease è un editor di testo scritto in tcl/tk. Può quindi essere usato sia su sistemi Windows che su sistemi Unix. È progettato per essere usato da programmatori e specialmente da chi sviluppa in ambiente multipiattaforma. Fra le caratteristiche: un menu contestuale per l'inserimento di commenti in svariati linguaggi di programmazione: sintax hilighting; supporto alle espressioni regolari; undo/redo illimitati; criptazione dei dati.

Directory /tease1.2.1

Marathon 0.90

Crea dei tutorial per le tua applicazioni Java

Si tratta di un software molto particolare: consente di catturare in un file di script tutte le azioni compiute durante l'esecuzione di un programma Java. Il file di script in questione può poi essere riutilizzato in un secondo momento per creare delle demo.

File: marathon-0.90-src.tar.gz

PMD 2.0

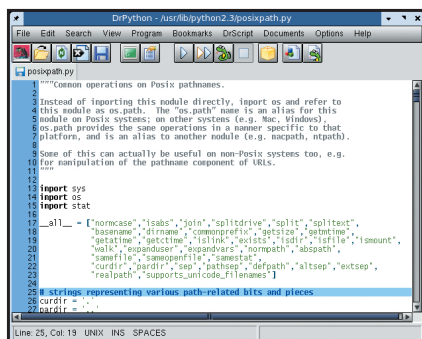
Un analizzatore di software scritti in JAVA, trova le variabili inutilizzate e molto altro

PMD scansiona il codice sorgente di un progetto scritto in Java alla ricerca di potenziali problemi quali ad esempio: blocchi *try/catch* vuoti, variabili locali inutilizzate, statements non necessari, loop basati su *for* che possono essere convertiti in cicli di *While* etc.
PMD ha plugin praticamente per ogni ambiente, si va da Eclipse, a Jedit a

Jbuilder, a Sun One Studio.
Directory: /PMD

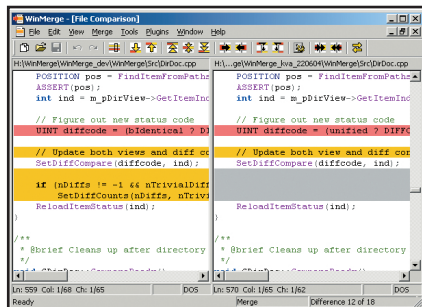
DrPython 3.6.10

Un editor per programmatori, scritto in Python

DroPyhtom è un editor altamente personalizzabile ed estendibile.
È scritto in Python ed è basato su wxPython. wxPython è un insieme di widgets utili per creare interfaccce grafiche in Python. Il textcontrol è invece basato su Scintilla.

Directory: \drpython

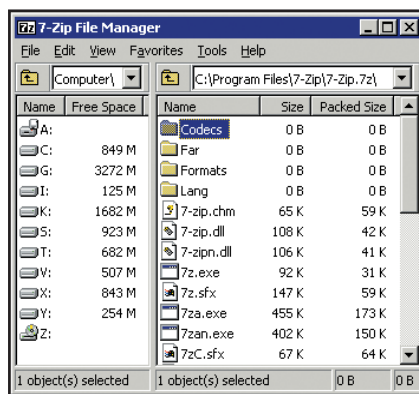
WinMerge 2.2

Effettua una comparazione di due file di testo anche di grandi dimensioni

WinMerge è un software Open Source che consente di effettuare il merging e il compare di due file di testo.
È utilissimo quando ad esempio si vogliono comparare le due diverse versioni di un file sorgente e capire cosa è stato modificato rispetto all'altra.

Directory: \winmerge

SevenZip 4.10

Uno dei programmi con migliore algoritmo di compressione per file.

Dotato ovviamente di sorgenti Sevenzip ha davvero un altissimo rapporto di compressione. Il fatto che i sorgenti siano disponibili e il programma basato su LPGL lo rende particolarmente interessante e adatto a essere studiato. Non di meno, utilizzare Seven ZIP porta sicuramente dei vantaggi. Supporta moltissimi tipi di file fra cui: 7z, ZIP, CAB, RAR, ARJ, GZIP, BZIP2, TAR, CPIO, RPM e DEB. Si integra perfettamente con la shell di Windows, è localizzato in 46 linguaggi ed ha una versione a linea di comando che lo rende adatto a essere usato per creare dei file batch.

Directory: /sevenzip

MaSal Editor 1.7.1649

Crea il tuo setup in un lampo

Un ambiente per la creazione di pacchetti di installazione che, per via grafica, consente di realizzare prodotti di livello professionale. Con apposite funzioni per il re-packaging, consente di aggiornare con facilità le vecchie versioni dei nostri software. La versione che trovate in allegato è gratuita, ma alcune funzioni sono disponibili solo nella versione a pagamento. È richiesta la presenza del .NET Framework 1.1.
masaieditor.exe

Setup Factory 7.0

Crea i file di installazione per distribuire le tue applicazioni!

Un sistema che rende semplicissimo costruire file di installazione per le applicazioni che sviluppiamo: settando le apposite opzioni, sarà poi facile distribuirli via Web, e-mail, FTP, CD-ROM o Lan. Per la creazione del pacchetto di installazioni, un wizard ci guida attraverso delle semplici azioni di drag & drop e, attraverso delle intuitive strutture condizionali, è possibile realizzare dei pacchetti di fattura altamente professionale e capaci di adattarsi alla macchina su cui si effettua l'installazione. Tra le funzioni supportate: installazione in un singolo *setup.exe*, per semplificare i down-

LIBRERIE

AlertPOP!

Messaggi popup in modo semplice e veloce.

[00024255.exe]

Xtreme Docking Pane

Finestre in stile Visual Studio.NET per le vostre applicazioni.

[00024164.exe]

r.a.d. panelbar

Menu a scomparsa dalla grafica accattivante generabili in pochi minuti.

[00021936.exe]

Block Messenger Spam

Un rimedio al mare di spam che invade i messenger service

[BLOCKMESS1617547192003.ZIP]

Esempi d'uso d'API

Giustappunto alcuni esempi d'utilizzo d'API

[AFEWUSEF128628992002.ZIP]

Janus WinForms Controls Suite

Una suite di controlli 100% codice .Net per una potente interfaccia stile Outlook.

[00024196.exe]

Xceed Ultimate Suite

Eccellenti controlli grafici ma non solo!

[00022095.exe]

Alphanumeric LED

Il necessario per impreziosire il vostro lavoro con riproduzione a LED di numeri e lettere.

[00021872.exe]

Alarm Clock

Un timer per far eseguire un determinato programma ad una certa ora.

[ALARMCLOC79811562002.ZIP]

LIBRERIE

Mail Checker 1.0
Come creare un programma per controllare la tua IMAP mail.
[csharpmailchecker10.zip]

Mail Reader
Un semplice SMTP mail client per inviare e ricevere mail.
[src_mailchecker.zip]

Springsys XsideBar per .NET
Una barra di controllo laterale simile a quella di Outlook.
[00022907.exe]

SharpUI per .NET
Per aggiungere una interfaccia utente grafica alle vostre applicazioni .NET
[00020122.exe]

A volte tornano...
Digger: un gioco "antico" ma reso ora attuale con .NET.
[DIGGER.ZIP]

Bar Chart in C#
Un semplice esempio di come creare dei diagrammi a barre in C#.
[BARCHART.ZIP]

DotNEtMagic-User Interface Library
Per avere un look stile Office2003 nei form delle vostre applicazioni.
[00023603.exe]

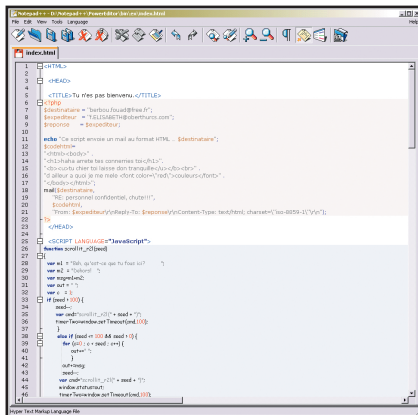
Probabiltà e statistiche
Statistiche, probabilità, distibuzioni e test d'ipotesi in una C# class library.
[00020559.exe]

Springsys Label3D
Effetti 3D per i testi delle tue pagine Web (ma non solo).
[00017599.exe]

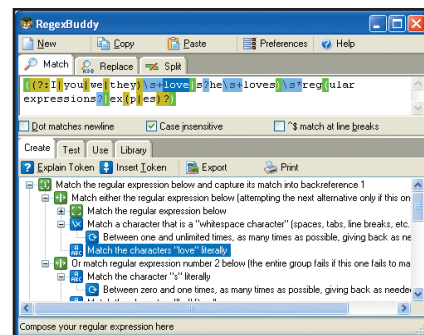
FLASH

Bar Graph Designer
Un'applicazione d'esempio che mostra come creare dei grafici a barre con flash
File: Bar_Grap-Yashraj_-8577.zip

load da Internet; verifica del numero seriale attraverso l'algoritmo crittografico MD5; controllo della data di scadenza per le versioni dimostrative; installazioni multi-lingua; scansione automatica dei progetti Visual Basic e molto altro ancora. Da segnalare il motore di compressione interno, particolarmente efficiente e rapido. Versione di valutazione valida ventuno giorni.
suf70ev.exe

Notepad++ 2.5
Un notepad col turbo!
Un editor gratuito (distribuito con licenza GPL) orientato alla manipolazione di codice e che offre il supporto a tutti i più diffusi linguaggi di programmazione: C, C++, Java, C#, XML, HTML, PHP, Javascript, file .ini, file batch, ASP, VB/VBS, SQL, CSS, Perl, Python, Fortran, actionscript e molti altri ancora. Il sintax highlighting è ottimamente realizzato e, attraverso le funzionalità di drag&drop, sarà semplicissimo utilizzare Notepad++ anche in progetti già avviati. Ampiamente personalizzabile: per i programmatori che non devono chiedere mai!

npp.2.5.Installer.exe

Eyebol Delphi Analyzer 1.20a
Analizza e ottimizza il tuo codice Delphi
Eyebol consente di analizzare i progetti Delphi al fine di ottimizzarne leggibilità ed efficienza.
Eyebol aiuta a rintracciare oggetti che non vengono rilasciati, va alla ricerca di codice e dati ridondandi e, attraverso 50 nuovi warning, consente di rendere più robusto il nostro codice.
Versione dimostrativa.
eyetry.exe

RegexBuddy 1.2.1
Crea e testa le espressioni regolari
Poche strumenti si sono rivelati così efficaci e "orizzontali" per i programmatori di qualsiasi piattaforma come le Regular Expressions: eppure non tutti riescono a impadronirsi subito del nuovo strumento espressivo. RegexBuddy si propone proprio come uno strumento per migliorare la nostra familiarità con le Reg Ex. Attraverso un'interfaccia di rara efficacia, oltre a testare direttamente le espressioni regolari su porzioni di testo importate con semplici drag&drop, potremo vedere una sorta di "spiegazione" in tempo reale dei singoli elementi che compongono la nostra espressione.
Utilissimo soprattutto nei casi in cui si voglia risalire al significato di espressioni regolari particolarmente lunghe e complesse.
Versione di prova valida sette giorni.

SetupRegexBuddyDemocnet-rxb-dl.exe

Java Launcher 1.5
Come lanciare applicazioni Java con un doppio clic
Un tool di rara semplicità che consente di distribuire applicazioni Java, garantendo all'utente la massima semplicità nell'avvio.
Java Launcher comprime tutte le classi e le risorse relative ad un'applicazione in un unico file *.EXE* che potrà essere lanciato come una comune applicazione Windows.
Gratuito.
javalauncher.zip

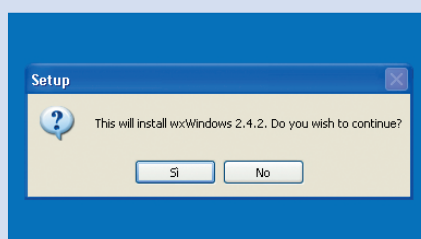
Python 2.3.4
Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà
Viene usato in una varietà di applicazioni. A quanto pare è largamente uti-

HELLO WORLD PYTHON

In questo box vi forniamo un minitutorial per l'installazione di Python e la realizzazione di una prima applicazione. Vi invitiamo a installare e provare questo linguaggio. Infatti, nonostante sia sconosciuto ai più, riteniamo che sia una delle piattaforme di sviluppo più potenti oggi in circolazione. Basti dire che, secondo quelli di Python Italia, si tratta di uno dei linguaggi base utilizzati per lo sviluppo di Google. Fra i suoi pregi principali c'è quello di avere una curva di apprendimento praticamente nulla, perciò si tratta di un linguaggio molto didattico, adatto anche a chi non ha basi solide di programmazione. Python vanta una sintassi derivata dal C, si tratta di un linguaggio ad oggetti molto elegante e molto potente, può essere utilizzato sia come piattaforma per applicazioni standalone o anche sul Web. In ultimo si tratta di un linguaggio multipiattaforma, funziona tranquillamente sia su sistemi Unix che su sistemi Windows, e anche su Mac anche nelle versioni precedenti alla 9.x. È un linguaggio interpretato, ciò porta a qualche lentezza di troppo, tuttavia il rapporto di righe di codice da scrivere a parità di applicazione fra Java e Python è di cinque a uno. Infine si tratta di una soluzione OpenSource.
Perciò vi invitiamo a prendere confidenza con questo strumento, e cominciare ad inviarci le vostre segnalazioni di interesse a *ioprogrammo@edmaster.it*. Noi riteniamo che sia un linguaggio che merita assolutamente di essere approfondito.

INSTALLIAMO PYTHON

1 **Il file da considerare è *Python-2.3.4.exe*. Il wizard è molto semplice. Al termine avremo gia tutto l'occorrente per scrivere la nostra prima applicazione. Invece preferiamo installare ancora qualcosa...**

INSTALLIAMO LE WXWIDGETS

2 **Il file da considerare è *wxMSW-2.4.2-setup.zip*. Per l'installazione è sufficiente seguire il wizard. Si tratta di librerie grafiche. Ci serviranno per DrPython, l'editor che useremo e per creare applicazioni in stile windows.**

WXWIDGETS FOR PYTHON

3 **Il file da considerare è *wxPython2.5-win32-unicode-2.5.3.1-py23.exe*. Si tratta delle librerie che fanno da ponte fra Python e le wxwidgets che abbiamo installato in precedenza.**

INSTALLIAMO DRPYTHON

4 **Il file da considerare è *drpython-3.6.9.zip*. Bisogna "scompattarlo" in una directory a piacere. Cliccando su *drpython.py* si avvierà l'editor. Naturalmente, possiamo scegliere un qualunque altro editor di testo.**

HELLO WORLD

```
print "Hello World"
print "welcome 2 python"
print "ioprogrammo"
```

5 **Digitiamo questo codice nell'editor e salviamolo in un file chiamato "*HelloWorld.py*". Dal prompt di MSdos digitiamo *HelloWorld.py* e, magicamente, ci compariranno le scritte che abbiamo inserito all'interno delle istruzioni Print.**

HELLO WORLD GRAFICO

6 **Copiate il codice nel box qui sotto, salvatelo in un file con estensione .py ed eseguitelo. Il risultato è quello in figura. Grazie alle *wxWidgets*, la formattazione dell'output è stata demandata a una semplice sintassi HTML.**

```
import sys
import wx # This module uses the new wx
                                namespace
import wx.html
import wx.lib.wxpTag
class MyAboutBox(wx.Dialog):
    text = '''
<html>
<body bgcolor="#AC76DE">
<center><table bgcolor="#458154"
                width="100%%" cellspacing="0"
        cellpadding="0" border="1">
<tr>
    <td align="center">
    <h1>Hello World</h1>
    <br>
    Welcome 2 Python <br><center>
                     ioProgrammo</center><br>
    </td>
</tr>
</table>

<p><wxp module="wx" class="Button">
    <param name="label" value="Okay">
    <param name="id"    value="ID_OK">
</wxp></p>
</center>
</body>
</html>
'''
    def __init__(self, parent):
        wx.Dialog.__init__(self, parent, -1,
                                 'Hello World',)
        html = wx.html.HtmlWindow(self, -1,
                                  size=(420, -1))
        if "gtk2" in wx.PlatformInfo:
            html.SetStandardFonts()
        html.SetPage(self.text)
        btn = html.FindWindowById(wx.ID_OK)
        ir = html.GetInternalRepresentation()
        html.SetSize( (ir.GetWidth()+25,
                         ir.GetHeight()+25) )
        self.SetClientSize(html.GetSize())
        self.CentreOnParent(wx.BOTH)

if __name__ == '__main__':
    app = wx.PySimpleApp()
    dlg = MyAboutBox(None)
    dlg.ShowModal()
    dlg.Destroy()
    app.MainLoop()
```

LIBRERIE

Draw Bezier Cubic

Come trascinare punti sensibili e disegnare curve di bezier

Bezier_C-nuran-8799.zip

PDA Board

Una piccola lavagna con tanto di color picker e strumenti per disegnare

Coachs_-Adam_Arm-9227.zip

Flash PHP hit

Un counter in Flash che si interfaccia a PHP per segnalare il numero di visite sulla pagina

Flash_Hi-Dean_Elz-9890.zip

Flash PHP hit 2

Ancora un counter, questa volta leggermente più complesso e sempre PHP come linguaggio di backend

flashph-jakub_ma-9421.zip

Framerate

Un piccolo esempio che mostra come variare in corso d'opera il Frame Rate di un filmato

Frame_sp-Tony_Hud-4227.zip

GPA Calculator

Una calcolatrice per valutare la media dei voti degli esami sostenuti

GPA_Calc-Farrukh_-8921.zip

Sfera di cristallo

Un effetto interessante che mostra come simulare il comportamento dell'acqua all'interno di una boccia trasparente

mystical-Lugia-9149.zip

Random N Generation

Genera un numero multiplo di uno in modo casuale

Random_n-fireman-8368.zip

Color Picker

Un bell'esempio di Color Picker, utile per imparare diverse tecniche

RGB_Colo-Marcus_B-8722.zip

Statistiche

Un'applet Flash collegata a PHP per monitorare il numero di accessi ad una pagina, anche mantenendo informazioni divise per mese

stats_fo-jose_jul-7994.zip

Ticker Maker 1.0

Decisamente carina questa applet che consente di creare al volo dei ticket come quelli dei cartelloni luminosi

TickerMa-Peter_Ba-9117.zip

lizzato addirittura dai tecnici di Google per lo sviluppo delle loro applicazioni.
Si caratterizza per la gestione dinamica della memoria, per l'elevata portabilità, per la curva di apprendimento relativamente breve. Un linguaggio di programmazione di cui sentiremo parlare a lungo.

Directory: \Python

Limnor 3.3

Un esperimento per creare applicazioni… senza codice!

Questo tool rappresenta un interessante esperimento che tenta di realizzare un ambiente per lo sviluppo di applicazioni che non utilizzi alcun linguaggio.
Attraverso un'interfaccia completamente drag&drop, Limnor consente di realizzare applicazioni indirizzate a chioschi interattivi e interfacce per CD-ROM.
Pur limitato a questi due ambiti, Limnor si dimostra molto flessibile e, attraverso un approccio object oriented, consente la creazione di applicazioni anche mediamente complesse.
Purtroppo, la scarsa efficacia dell'interfaccia rende alquanto farraginoso il compito di costruire le applicazioni. Merita comunque un'occhiata.

limnor.msi

DATABASE

HSQL Database engine 1.7.2

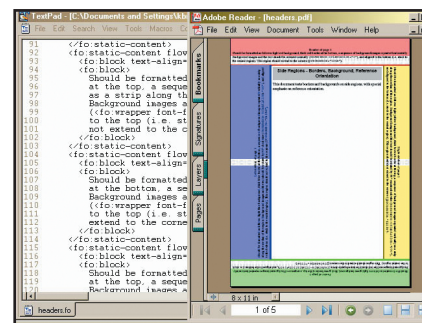
Un database relazionale scritto in Java

HSQLDB può essere utiilizzato tranquillamente tramite JDBC. Fra le sue caratteristiche più interessanti è che ne esiste una versione di appena 100k, nonostante sia un engine di database molto veloce che può gestire i dati sia usando la memoria fisica sia il disco.
Addizionalmente comprende un Web Server e tool visuali di gestione.

Directory: \hsqldb

XEP 4.0

Per connvertire documenti XML in PDF o PostScript

Un'applicazione commerciale in grado di effettuare la conversione da documenti XML in PDF o in formato PostScript. Accetta in input dati in formato XML e fogli di stile XSL, il rendering è effettuato proprio come combinazione dei due ingressi. XEP offre un valido supporto a XSL FO, grafici SVG e funzioni di link PDF-to-PDF. Richiede sia installata una Virtual Machine Java.
Tra le migliorie introdotte con la versione 4.0, si segnala la possibilità di ridimensionare dinamicamente le tabelle presenti nei documenti.
Versione dimostrativa, un watermark è applicato su ogni documento prodotto.

setup-4.0-trial.zip

W2XML 2.3

Per convertire documenti Word, RTF e HTML in XML

Un accessorio indispensabile nella cassetta degli attrezzi di uno sviluppatore: questo tool consente di convertire in modo automatico e veloce i file *.doc* in formato XML. Gli stili di Word sono riportati nel formato di destinazione e, grazie ai documenti XSLT personalizzabili, è possibile sfruttare anche le informazioni associate allo stile. Tra le nuove funzionalità si segnala la generazione automatica di un file *css* per l'utilizzo immediato dei documenti XML in ambito Web. La conversione è molto rapida e risulta estremamente utile la possibilità di convertire gruppi di file in un colpo solo. Ampiamente personalizzabile, può essere adattato alle più disparate esigenze. Per una corretta installazione, richiede che sia stato precedentemente installato il .NET Framework ed il pacchetto Universal Application Console, quest'ultimo riportato sul CD.
Versione di valutazione della durata di quindici giorni, limitata alla conversione di 15 documenti.

W2XML23Trial.msi

INSTALLARE BUGZILLA IN AMBIENTE WINDOWS

In questo tutorial daremo per scontato che abbiate già installato una versione funzionante di MySQL e una versione funzionante di Apache. Fatto questo i passi successivi prevedono:

1 **Installare Perl dal sito *ActiveState* http://www.activestate.com. Stiamo lavorando per distribuire una versione dell'*ActivePerl* sul CD allegato ad ioProgrammo, ma purtroppo non abbiamo ancora ricevuto l'*ok* da parte di *ActiveState*, quindi al momento siete costretti a scaricarlo dal sito di *ActiveState*.**

2 **Scompattare il file *bugzilla-2.18rc3 .tar.gz* presente nel CD di ioProgrammo e copiare i file nella directory *htdocs* appropriata**

3 **Lanciare un prompt di MS Dos e portarsi nella directory dove risiede l'installazione di Bugzilla. Molto probabilmente *htdocs\bugzilla***

4 **Dal prompt di MS Dos lanciare il comando *checksetup.pl*. Il comando in questione rileverà se la vostra installazione di Perl è compatibile con l'installazione di Bugzilla oppure se mancano dei moduli. Molto probabilmente mancheranno dei moduli. Sarà bugzilla stesso ad avvertirci di quali moduli mancano e come fare a installarli. In particolare dovremo lanciare il comando**

```
> ppm rep add bugzilla
            http://landfill.bugzilla.org/ppm/
```

e poi installare i moduli che mancano:

```
> ppm install nome modulo
```

Una volta installati tutti i moduli passeremo al passo 5

5 **Lanciare ancora una volta il comando *checksetup.pl*. Verrà generato un errore che ci avverte che non è configurato mySQL, niente di grave, verrà anche generato un file "*localconfig*" da editare per inserire i parametri corretti di connessione a mysql. In particolare:**

```
# How to access the SQL database:
#
$db_host = "localhost"; # where is the
                                database?
$db_port = 3306;  # which port to use
$db_name = "bugs"; # name of the MySQL
                                database
$db_user = "bugs"; # user to attach to the
                           MySQL database
$db_pass = '';
```

6 **Creiamo il database con**

```
mysqladmin.exe -u root -ppasswd create bugs
```

sostiduendo a *Passwd* la vostra password

7 **Connettiamoci a mysql come utenti root con**

```
mysql -uroot -plavostrapassword
```

e dalla shell di mysql settiamo i permessi e la password dell'utente bugs

```
mysql> grant all on bugs.*
       to bugs@localhost identified by 'bugs';
```

il che fa sì che un utente *bugs* con password *bugs* abbia diritto di connettersi al database *bugs* su MySQL

8 **Modifichiamo il file "*localconfig*" per i nostri scopi**

9 **Lanciamo *checksetup.pl* e rispondiamo alle poche domande che ci verranno fatte, come ad esempio l'indirizzo email dell'amministratore del sistema**

TRUCCHI FINALI IN AMBIENTE WINDOWS

Dobbiamo sopperire alla mancanza del *SendMail*, perciò in *Bugizilla/Bugmail.pm* possiamo sostituire le linee

```
open(SENDMAIL, "|/usr/lib/sendmail
        $sendmailparam -t -i") ||die "Can't
                          open sendmail";
print SENDMAIL trim($msg) . "\n";
close SENDMAIL;
```

Con

```
use Net::SMTP;
my $smtp_server = 'smtp.mycompany
                .com';  # change this
# Use die on error, so that the mail will be
            in the 'unsent mails' and
# can be sent from the sanity check page.
my $smtp = Net::SMTP->new(
   $smtp_server) || die 'Cannot  connect to
                server \'$smtp_server\'';
$smtp->mail('bugzilla-daemon@
        mycompany.com');  # change this
$smtp->to($person);
$smtp->data();
$smtp->datasend($msg);
$smtp->dataend();
$smtp->quit;
```

CONFIGURAZIONE DI APACHE

È sufficiente aprire il file *httpd.conf* e aggiungere le seguenti righe, sostituendo chiaramente il percorso corretto

```
AddHandler cgi-script .cgi
<Directory "C:\Programmi\Apache Group\
             Apache\htdocs/bugzilla">
Options +ExecCGI +FollowSymLinks
 AllowOverride Limit
 Options Indexes
 DirectoryIndex index.cgi
</Directory>
```

A questo punto tutto dovrebbe funzionare, e invece sorge un problema. Dopo avere effettuato il restart di Apache, se siete in ambiente Windows vi ritroverete davanti al seguente errore:

Internal Server Error
The server encountered an internal error or misconfiguration and was unable to complete your request.

Il problema sta nel fatto che nel software di Bugzilla nella prima riga di ogni file *.cgi* compare un'intestazione del tipo

```
#!/usr/bin/perl -wT
```

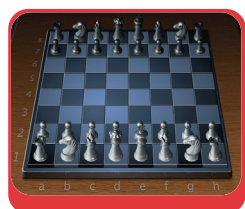
che indica la posizione dell'interprete perl con cui la pagina deve essere generata.
In ambiente Windows questo chiaramente non può funzionare! Siamo costretti a sostituire in ogni file del progetto la riga di intestazione con

```
#!c:/Perl/bin/Perl.exe
```

oppure con il percorso equivalente del perl sul vostro sistema.
Poichè personalmente non avevo nessuna intenzione di compiere manualmente questa operazione, ho usato il fantastico *Batch Replacer Btr Wizard*, presente in questo stesso numero di ioProgrammo.

Costruzioni matematiche dalla storia ricca e interessante

Introduzione ai quadrati magici

Si tratta di matrici quadrate di numeri dalle singolari caratteristiche Magicamente le somme: per riga, per colonna e sulle due diagonali danno sempre lo stesso numero

Utilizza questo spazio per le tue annotazioni

Un quadrato magico è una griglia quadrata (con eguale numero di elementi per riga e per colonna) che gode di straordinarie peculiarità rispetto all'operazione di somma. Prendendo i numeri di una qualsiasi riga e sommandoli, il risultato ottenuto è un nuovo numero che si ripresenta per tutte le somme che si compongono considerando una qualsiasi altra riga, una qualsiasi colonna o una delle due diagonali. Un banale quadrato magico è definito da nove elementi incasellati in una griglia di dimensione tre righe e tre colonne:

6	1	8
7	5	3
2	9	4

Si può notare come la somma per le tre righe, le tre colonne e le due diagonali faccia sempre lo stesso numero 15. A proposito di 15, ricordate il gioco omonimo esaminato due numer fa? Aveva richiamato intrinsecamente la trattazione dei quadrati magici. Ecco il contatto, il sottile filo conduttore che indica il percorso da seguire nelle nostre esplorazioni ludico-scientifiche. In generale per una fissata dimensione vi sono più soluzioni; per la tre per tre ecco un'altra soluzione.

6	7	2
1	5	9
8	3	4

DEFINIZIONE E PROPRIETÀ

La dimensione della matrice, detta ordine del quadrato magico, è solitamente indicata con n. Un altro numero molto importante è la somma per generica riga, colonna e diagonale, che per definizione dei quadrati rimane invariante; tale valore è conosciuto come costante magica e verrà indicato con m. Esiste una fondamentale relazione che lega i due numeri "cardine" che descrivono i quadrati magici: l'ordine e la costante magica. L'espressione matematica è la seguente:

$$m = \frac{n^3 + n}{2}$$

Quindi la somma dei numeri per riga, colonna o diagonale non può essere una nostra scelta ma deve appunto soddisfare l'appena scritta relazione. Risulta, inoltre, che i numeri che costituiscono il quadrato magico possono appartenere al range *[1..n2]* sempre qualora siano degli interi. L'ultima precisazione è d'obbligo perché alcune varianti dei quadrati magici considerano numeri con reali o razionali (ovvero con virgola). Un'altra importante classificazione distingue quadrati magici dispari e pari.
I primi sono caratterizzati dall'avere n dispari, per i secondi l'ordine è pari; questi ultimi quadrati magici sono più difficoltosi da individuare. Gli algoritmi di costruzione sono infatti più complessi.
I quadrati magici rispettano alcune proprietà; ecco le più importanti:

- Un quadrato magico rimane tale se un qualsiasi numero viene addizionato ad ogni elemento del quadrato;
- Un quadrato magico rimane tale se un qualsiasi numero moltiplica ogni elemento del quadrato;
- Un quadrato magico rimane tale se due qualsiasi colonne o righe equidistanti dal centro vengono scambiate;
- Un quadrato magico di ordine pari rimane tale se i quadranti opposti vengono scambiati (i quadranti sono quattro e sono i quadrati di ordine n/2 ottenuti dalla divisione nell'esatta metà delle righe e delle colonne);
- Un quadrato magico di ordine dispari rimane

tale se i quadranti parziali opposti vengono scambiati (si parla di quadranti parziali poiché, rispetto al caso pari, non bisogna computare la riga e la colonna centrale, essendo dispari il numero di righe e di colonne).

Alcune delle proprietà vengono usate per ottenere gli algoritmi di costruzione.

LA STORIA DEI QUADRATI MAGICI

La pagina storica riferita ai quadrati magici è consistente. Anche i personaggi che nel corso degli anni si sono interessati, anche solo per gioco, della questione sono nomi conosciuti nell'ambito scientifico. Se a questo si aggiunge che il gioco ricopre una valenza artistica- esoterica e culturale-scientifica si comprende il largo consenso riscontrato nel corso dei secoli. Cercherò di limitare i miei impulsi appellandomi alle doti di sintesi, visto che sono molto affascinato al particolare aspetto storico; sarei quindi tentato di fare una vera e propria trattazione. Il primo quadrato magico di cui si ha traccia affonda la sua storia tra leggenda e realtà. Si tratta di "Lo Shu" (il saggio del fiume Lo) risalente a circa 4200 anni fa. Si narra che l'imperatore Yu, della dinastia Hsia, mentre si trovava in meditazione sul fiume Giallo, dalle acque vide apparire una tartaruga. Sul dorso dell'animale, formato da caselle, e assimilabile quindi ad una matrice, vi erano dei simboli incasellati nei naturali quadrati descritti dalla corazza. In studi che risalgono a circa il 400 A.C. sono stati esaminati i segni; si è riscontrato che, la somma sei singoli segni, secondo le regole che conosciamo, descriveva un quadrato magico. Si ritrova un quadrato magico di tipo simmetrico nella famosa incisione di Albrecht Durer "La malanconia"; il particolare è riportato in **Figura 1**. In Europa pare che siano stati introdotti da E. Moscopulo verso il 1420, ma solo quelli di ordine dispari, che però i matematici indiani già sapevano costruire. Nel Medioevo furono attribuite virtù soprannaturali e vennero usati come amuleti, successivamente furono introdotti in astrologia. Tra i numerosi studiosi che affrontarono l'argomento ricordo: C.G. Bachet, L. Eulero, A.H. Frost, G Arnoux e B. Franklin (famoso per l'invenzione del parafulmine); il suo quadrato è di ordine 8.

	1	2	3	4	5	6	7	8	9
1	18199	20533	15901	18229	19417	17293	17827	29983	6571
2	15913	18211	20509	17377	18313	19249	6871	18127	29383
3	20521	15889	18223	19333	17209	18397	29683	6271	18427
4	18097	20479	15823	17851	33073	3727	18289	36583	31
5	15859	18133	20407	4093	18217	32341	43	18301	36559
6	20443	15787	18169	32707	3361	18583	36571	19	18313
7	17977	32917	4027	18061	21649	14653	18013	31543	5113
8	4357	18307	32257	14713	18121	21529	5323	18223	31123
9	32587	3697	18637	21589	14593	18181	31333	4903	18433

Tabella 1: Una matrice 9x9 che rappresenta un quadrato magico

CONCLUSIONI

Il quadrato proposto in apertura è ovviamente magico, ha però un'interessante proprietà aggiuntiva; ogni numero è primo. Data la sua grandezza e il "pesante" vincolo della primalità di tutti gli elementi si tratta di un quadrato da incorniciare. Esistono anche altri quadrati che alle caratteristiche standard ne aggiungono di nuove, essi prendono nomi come quadrati supermagici. Saranno oggetto delle nostre osservazioni nei prossimi numeri.

Fabio Grimaldi

Fig. 1: Particolare dell'opera "La malanconia" di A. Durer quadrato magico simmetrico

L'ANGOLO DELLA COMPETIZIONE

In occasione della prima fase di selezione per le olimpiadi di informatica, tenutasi presso i singoli istituti scolastici che hanno aderito alla competizione migliaia di studenti si sono mobilitati per ottenere un buon risultato. Da buon professore di informatica (almeno spero di essere considerato "buon") renderò partecipi i lettori, nel corso degli appuntamenti di enigma, di alcuni dei problemi comparsi. Riemerge, quindi, lo spazio sulla rivista dove gli appassionati del genere potranno testare il proprio quoziente di capacità informatiche. Ricordo che gli esercizi somministrati sono di due generi, e sono orientati a valutare le capacità logiche e di programmazione. Partiamo con due semplici problemi di logica che valgono entrambi tre punti.

1. **Si consideri un torneo di calcetto in cui ogni squadra deve incontrare esattamente una volte tutte le altre. Se il numero di partite è 136, quale è il numero delle squadre?**
2. **Si consideri un ipotetico gioco di realtà simulata che si effettua su un campo quadrato diviso in 6 righe e 6 colonne (contenente quindi un numero di caselle uguale a 36). All'inizio del gioco alcune caselle possono essere focolai di epidemie. Le epidemie si diffondono secondo il seguente schema: una casella viene infettata quando è adiacente ad almeno due delle caselle infette (due caselle sono considerate adiacenti quando condividono un lato; per esempio, due caselle vicine in diagonale non sono adiacenti in quanto condividono un solo punto).
Quale è il numero minimo di caselle focolari di epidemia capaci di infamare tutto il campo da gioco?**

Nel prossimo numero daremo le soluzioni ai quesiti e ne proporremo di altri (anche di programmazione).

Metodi iterativi per la simulazione di sistemi naturali

Sistemi reali simulati

In questo articolo analizzeremo un metodo matematico per costruire immagini molto speciali partendo da una semplice stringa Regole e assiomi per disegnare frattali

Utilizza questo spazio per le tue annotazioni

Nell'ampio panorama delineato dai sistemi caotici si distinguono per efficienza e potenzialità implementativa i sistemi di Lindermayer, che indicheremo semplicemente come l-system. Nello scorso appuntamento è stata aperta una grande finestra sul mondo dei sistemi caotici. Nella vasta classificazione approntata, abbiamo segnalato diversi modi per affrontare l'argomento. Abbiamo potuto notare come gli approcci sono molto diversi tra loro. Esplorando ancora questo ambito, che tanto ci piace per le piacevoli applicazioni di programmazione che si possono realizzare, visioneremo nuovi metodi, ma soprattutto ne approfondiremo qualcuno. In questo articolo ci dedicheremo esclusivamente al metodo *l-system* cercando di scoprirne gli arcani e proponendo soluzioni innovative. Non solo: l'argomento ci riserverà ancora aspetti di interesse e valore, ad esempio la rappresentazione grafica delle stringhe prodotte.

ASSIOMA E PRODUZIONE

Il metodo si basa su un semplice processo di produzione. A partire da due elementi: un stringa iniziale detta assioma e una regola di produzione, così facendo si ottengono i risultati e si può studiare l'evoluzione del sistema caotico. L'assioma è il punto di partenza, la prima componente su cui si applica l'intero processo di produzione. La regola, o l'insieme delle regole di produzione, vanno applicate per generare nuovi risultati. L'evoluzione avviene in modo discreto, non intesa nel senso che non disturba nessuno, bensì che progredisce a intervalli finiti di tempo. Infatti, si parlerà di generazione. Applicando la regola di produzione all'assioma si ottiene la produzione della prima generazione. Se il risultato ottenuto viene nuovamente sottoposto alle regole di produzione si genera la seconda generazione, e così via. Osservando una qualunque generazione, si possono fare delle valutazioni sugli effetti che le regole imprimono sulla popolazione (stringa) iniziale.
Tutto ciò ci porta a due considerazioni. La prima rileva una forte analogia con alcuni processi di evoluzione, che sono propri di analisi proposte tra queste pagine, come gli automi cellulari e il gioco della vita. La seconda ci ricorda alcune applicazioni del metodo, che spaziano dalla semplice raffigurazione grafica per la realizzazione di piacevoli effetti visivi, all'utilizzo che ne viene fatto in campo bio-matematico in cui il metodo simula la crescita e l'evoluzione di forme di vita e fenomeni naturali. Per capire il metodo riprendiamo il semplice esempio accennato la scorsa volta. Bisogna definire due elementi l'assioma, e la regola di produzione; essi sono riportati qui di seguito:

F+F+F+F (assioma)
F -> F+F-F-FF+F+F-F (regola di produzione)

Applicando la produzione all'assioma si dà origine alla prima generazione. Si tratta di sostituire ogni elemento F con la stringa proposta nella regola e lasciare inalterati gli altri simboli (nel caso specifico sono presenti i soli simbolo + e -). Così, il primo risultato è:

F+FF-F+F+F+FF-F+FF+FF-F+F-F+FF-F+F+F+FF-F+F+F+FF-F+F+F+FF-F+FF+FF-F+F-F+FF-F+F+F+FF-F+F

Il processo può essere iterato. È chiaro che ad ogni nuova generazione si otterranno stringhe di lunghezza sempre maggiore. Il particolare l-system creato come esempio, si presta ad un'interessante interpretazione grafica. Bisogna associare significato ai simboli. Per esempio, F significa tracciare una linea in avanti (lunghezza e spessore sono specifiche da indicare in fase di realizzazione); il simbolo + significa cambia direzione di 90 gradi (anche l'ango-

lo può essere un elemento variabile); mentre – gira a sinistra di 90°. La produzione applicata al simbolo F sostituisce un singolo segmento con la **Figura 1a**, come si può osservare esaminando il significato dei singoli simboli. La successiva applicazione della regola sull'assioma (che come si può osservare è un quadrato, quattro segmenti posti di seguito con angolo, tra ognuno, di 90°) produce la stringa più corposa ottenuta sopra. La sua rappresentazione grafica è simile ad un pezzo di costruzioni a incastro; è riportata in figura 1b. Iterando ancora il processo si produce la figura 1c. Come si può notare sono rispettate le caratteristiche di autosimilarità proprie dei frattali. Una generalizzazione del metodo prevede che ci possano essere più regole di produzione. Inoltre, a ogni simbolo si può associare un significato, così da simulare forme di vita (come alberi). Le stringhe prodotte sono facilmente rappresentabili graficamente; a tale scopo risulta utile costruire una tabella di corrispondenza simbolo-significato. In **Tabella 1** è mostrato un semplice esempio. Ovviamente, la tabella può essere anche più ricca di simboli e dei corrispondenti significati. In **Figura 2** è proposto un l-system descritto da due regole di produzione e dal dall'assioma A, e la sua corrispondente rappresentazione grafica.

UNA PRIMA SOLUZIONE

La prima soluzione proposta è poco flessibile e ha alcuni limiti implementativi, ha però il pregio di essere chiara. Si tratta quindi di un ottimo esempio per cominciare. Si vuole produrre il sistema riportato in **Figura 1**. A tale scopo si predispone una stringa *s* che indica l'assioma e che conterrà nelle varie fasi dell'esecuzione le varie produzioni. Con le due variabili *n* e *g* indichiamo rispettivamente il numero di generazioni che si intende produrre e la popolazione (stringa) corrente.

```
(* lsyst.pas
   prima soluzione *)
Program LSystem;
const smax = 255;
type tstr=string[smax];
var s,ss : tstr;
     n,g : integer;
procedure produzione (vs:tstr; var ns:tstr);
var i:integer;
Begin
        for i:=1 to length(vs) do
        case vs[i] of
        'F':ns:=ns+'F+F-F-FF+F+F-F'
        else ns:=ns+vs[i];
        end
end;
Begin (* main *)
        s:='F+F+F+F';
        write('N. rig. --> ') ;
        readln(n);
        for g:=1 to n+1 do
        Begin
          writeln(s);
          writeln;
          ss:=s;
          s:='';
          produzione(ss,s);
        End;
End.
```

Nel main program si può scorgere il ciclo dove si producono le nuove stringhe, qui la stringa ss contiene l'assioma e s la produzione. La sostituzione della vecchia stringa vs (assioma) con la nuova stringa *ns,* avviene nella procedura produzione. Eseguendo il programma potrete notare come si produca la stringa prima proposta in esempio. Tale soluzione soffre due importanti limiti. Innanzitutto, è poco flessibile attua la sola produzione della regola *F -> F+F-F-FF+F+F-F,* quindi è statica; inoltre ha limiti di memoria visto che in Pascal (ma anche in molti altri linguaggi) le stringhe possono avere una lunghezza massima di 256 caratteri. Occorre una nuova soluzione.

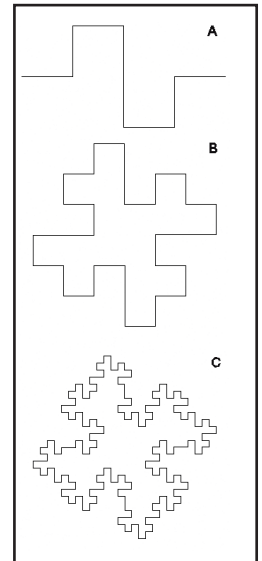



Fig. 1: ***l-system nelle prime tre fasi di generazione***

SOLUZIONE PIÙ EFFICIENTE

Nel progettare una nuova soluzione si deve tenere conto dei limiti della precedente. Il nuovo programma deve prevedere la possibilità: di produrre stringhe di lunghezza maggiore a 256; di fare riferimento a un insieme di regole e non limitarsi ad una soltanto; di inserire le regole da input così da rendere il codice più flessibile. La soluzione di seguito risponde a tutti i requisiti indicati. Per il soddisfare il primo si è reso necessario cambiare la struttura dati. È stata infatti, implementata una lista a puntatori e un vettori di tali liste, che contengono le singole stringhe (ogni

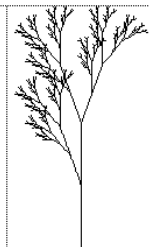



Fig. 2: ***l-system applicato alla simulazione della crescita di una pianta***

Simboli	Significato
A	Muovi in avanti tracciando un segmento
+	Gira a sinistra di angolo° (angolo è una costante)
-	Gira a destra di angolo° (angolo è una costante)
>	Moltiplica la lunghezza della linea di un fattore di scala
<	Dividi la lunghezza della linea di una di un fattore di scala
[	Fai una push del corrente disegno, usando un apposito stack
]	Fai una pop del disegno

Tabella 1: I valori attribuiti alla tabella

elemento è un carattere), in modo da non aver nessun limite se non quello della memoria riservata al compilatore. Ciò ha richiesto alcune accortezze che personalmente mi hanno stimolato; sono sicuro che i lettori (programmatori) apprezzeranno l'implementazione. Sono previste un massimo di dieci regole di produzione a cui sono state associate le prime dieci lettere maiuscole dell'alfabeto. Di seguito è riportato l'intero programma pronto a essere compilato. Su cd troverete anche l'eseguibile associato.

```
Program LSystem;
(*massimo numero di regole e generazioni*)
const gmax=10;
(* I tipi definiti dall'utente:
   - lista a puntatori (tpunt);
   - vettore lista a puntatori (tvetpunt) *)
type  tpunt=^nodo;
      nodo=record
            c:char;
         next:tpunt;
         end;
      tvetpunt=array[1..gmax] of tpunt;
var
      (* Assiomi (s) e regole di produzione (pr) *)
       pr,s : tvetpunt;
       (* numero di generazioni (n),
          generazione corrente (g) *)
        n,g : integer;
```

La parte dichiarativa comprende i tipi prima descritti nonché le variabili globali che hanno lo stesso significato del programma precedente. I due vettori di liste a puntatori s e pr contengono rispettivamente i generici assiomi e le regole di produzione. Le prime tre procedure si occupano: della inizializzazione del vettore di stringhe che conterrà gli assiomi; del caricamento di una stringa su una generica lista; e infine del caricamento del set di regole su un vettore di liste. Questa ultima procedura si avvale della precedente per il trasferimento di una singola lista. Il caricamento avviene scorrendo al contrario sulla stringa (dall'ultimo carattere al primo). Poiché si attua un inserimento in testa che simula una pila, ovvero, l'ultimo elemento si trova come primo, allora la lista, in definitiva, si trova nel giusto verso.

```
(* Inizializzazione del vettore di puntatori *)
procedure config(var v:tvetpunt);
var i:integer;
Begin
      for i:=i to gmax do
          v[i]:=nil
End;
(* Carica una stringa (mess) su un puntatore (p) *)
procedure carica(var p:tpunt; mess:string);
var stemp:string[30];
    ptemp:tpunt;
        i:integer;
begin
        write(mess,' --> ');
        readln(stemp);
        new(p);
        p:=nil;
        for i:=length(stemp) downto 1 do
        begin
          new(ptemp);
          ptemp^.c:=stemp[i];
          ptemp^.next:=p;
          p:=ptemp;
        end;
end;
(* Carica un insieme di regole sul vettore delle regole
                                              di prod.*)
procedure caricpr(var produz:tvetpunt);
var   i,m:integer;
        r:char;
  stringa:string[20];
Begin
        config(produz);
        writeln('Carica le regole di produzione');
        write('N. di regole da caricare : ');
        readln(m);
        r:='A';
        for i:=1 to m do
        begin
              (* per caricare la singola regola *)
              carica(produz[i],r);
              r:=chr(ord(r)+1)
        end;
End;

(* alloca nuova memoria dinamica ricopiando pold su pnew *)
procedure copia(var pnew:tpunt; pold:tpunt);

var pp,pfine:tpunt;
Begin
      (* inserimento in coda *)
      if pold <> nil then
      begin
          new(pp);
          pp^.c:=pold^.c;
          pp^.next:=nil;
          pfine:=pp;
          pnew:=pp;
          pold:=pold^.next;
      end;
      while pold<>nil do
      begin
          new(pp);
          pp^.c:=pold^.c;
          pp^.next:=nil;
          pfine^.next:=pp;
          pfine:=pp;
          pold:=pold^.next;
```

COMPILATORE FREE

Il codice è stato compilato con free pascal una bella sorpresa nel panorama dello sviluppo libero. Il compilatore scaricabile da www.bloodshed.net è davvero molto potente e simula sia l'ambiente tipico degli storici compilatori dos di Borland, sia ambienti agganciati alle piattaforme evolute della famiglia Windows.

```
   end;
End;
(* Appende la lista pcar in coda alla lista p *)
procedure aggiungi(var p:tpunt; pcar:tpunt);
var pp,pcarcopy:tpunt;
Begin
    (* alloca nuova memoria dinamica ricopiando la
                          parte da appendere *)
    copia(pcarcopy,pcar);
    pp:=p;
    if p=nil then  p:=pcarcopy
    else
    begin
    while pp^.next<>nil do
      pp:=pp^.next;
    pp^.next:=pcarcopy;
    end;
End;
(* Visualizzazione di una singola lista *)
procedure output(p:tpunt);
Begin
    while p<>nil do
    begin
    write(p^.c);
    p:=p^.next;
    end;  writeln;

End;
(*Applicazione delle regole di produzione (vettore ppr)
  alla vecchia lista (vs); producono la nuova lista (ns) *)
procedure produzione (var ns:tpunt; vs:tpunt;
                                   ppr:tvetpunt);
var
   i:integer;
   scan:tpunt;
Begin
    while vs<>nil do
    begin
    new(scan);
    scan^.c:=vs^.c;
    scan^.next:=nil;
    case scan^.c of
    'A': aggiungi(ns,ppr[1]);
    'B': aggiungi(ns,ppr[2]);
    'C': aggiungi(ns,ppr[3]);
    'D': aggiungi(ns,ppr[4]);
    'E': aggiungi(ns,ppr[5]);
    'F': aggiungi(ns,ppr[6]);
    'G': aggiungi(ns,ppr[7]);
    'H': aggiungi(ns,ppr[8]);
    'I': aggiungi(ns,ppr[9]);
    'J': aggiungi(ns,ppr[10]);
    else aggiungi(ns,scan);
    end;
    dispose(scan);
    vs:=vs^.next
    end;
end;
```

La procedura fondamentale è produzione che si occupa di sostituire un assioma con una nuova stringa, applicando il set regole. Il funzionamento è semplice.
Si scorre l'assioma è si valutano tutti i simboli ad uno a uno con il classico parser. Ogni elemento viene confrontato con i simboli a cui è associata una regola di produzione (nell'istruzione case), qualora si individui una corrispondenza si aggiunge, con l'omonima procedura, la regola di produzione alla stringa in via di costruzione; altrimenti, il simbolo non riconosciuto (ad esempio il + oppure il -) indicato con scan viene ricopiato così come è. Nella fase di append è comunque necessario allocare nuova memoria per creare le nuove liste. Questo aspetto è demandato alla routine copia. La procedura output visualizza una singola lista. Infine, il programma principale dopo aver inizializzato e caricato gli elementi di input mediante un ciclo produce g generazioni di l-system.

```
Begin (* main *)
    writeln;
    writeln('-- Lindermayer System --');
    writeln(' # _ una realizzazione de ilmagodifibra #');
    writeln(' * implementazione: liste a puntatori *');
    writeln;
    config(s);
    carica(s[1],'Assioma');
    caricpr(pr);
    write('N. rig. --> ') ;
    readln(n);
    for g:=1 to n do
    Begin
      produzione(s[g+1],s[g],pr);
      writeln(g+1,' > ');
      output(s[g+1]);
      writeln;
    End;
End.
```

CONCLUSIONI

Il programma sviluppato può essere migliorato in diverse direzioni. Una prima funzionalità che va aggiunta è la possibilità di archiviare su file una produzione; e la operazione inversa di caricamento di un l-system da file. Si può, inoltre, pensare a una specifica interpretazione grafica di una generica stringa previa la costruzione di una tabella di corrispondenza simbolo-significato grafico. Questi sono alcuni degli interrogativi che ci porremo nel prossimo numero per completare al meglio un programma per la simulazione di un l-system. Scopriremo che le applicazioni sono davvero numerose.
Alla prossima!!

Fabio Grimaldi

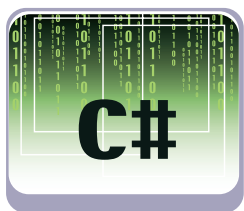
ALCUNI SISTEMI SIMILI

Nell'ambito dei sistemi caotici sono conosciuti gli IFS (Iterated function system) che hanno alcune similarità con gli l-system, infatti, producono rappresentazioni grafiche simili. Tali sistemi sostituiscono poligoni con altri poligoni. Ad ogni iterazione i poligoni sono scalati, ruotati e traslati.

Le caratteristiche più interessanti di Visual Studio Express 2005

Visual C# 2005 Express Edition

Visual C# 2005 Express Edition beta è liberamente scaricabile da Microsoft. Scopriamo come possiamo utilizzare le nuove caratteristiche per migliorare la produttività

Utilizza questo spazio per le tue annotazioni

Disponibili come download gratutiti su *msdn*, le versioni beta di Visual Studio Express 2005 sono un chiaro segno della strategia di Microsoft per quanto riguarda lo sviluppo su .NET dei prossimi anni.
Allargare la base di sviluppatori anche a coloro che sono abituati all'utilizzo di strumenti free e open source e nel contempo far testare sul campo alla comunità di sviluppatori .NET le nuove caratteristiche di C#2 e dell'IDE di Visual Studio 2005.
Per ora è possibile scaricare gratuitamente le versioni per lo sviluppo winforms e web di VS Express 2005 sia per C# che per VB.NET.
Microsoft ad oggi non ha svelato quale sarà il modello di licenza della release ufficiale di VSExpress 2005, ma sicuramente sarà un prodotto a basso prezzo, accessibile da un vasto bacino di utenti.
In questo articolo vedremo alcune delle caratteristiche più interessanti del nuovo ambiente di sviluppo, IDE che nelle intenzioni dei progettisti Microsoft dovrebbe aumentare la produttività e migliorare l'esperienza degli sviluppatori.
Molte delle nuove caratteristiche di VS2005 sono presenti nella versione Express.
Possiamo dividere le novità secondo l'utilizzo, ovvero quelle che riguardano più direttamente l'aiuto alla stesura del codice e quelle di infrastruttura.

STESURA DEL CODICE

Cominciamo questa nostra carrellata dalle caratteristiche che permettono una più agile stesura del codice. In particolare abbiamo un Intellisense più sviluppato e un insieme completo di funzionalità legate al refactoring del codice.

Expansions *(espansioni)*
Nella pratica quotidiana di sviluppo esistono strutture logiche che vengono ripetute in maniera pressappoco simile. Si pensi ai cicli *for*, o *while*. Le *Expansions* forniscono delle strutture precompilate in cui lo sviluppatore deve solo scrivere le parti variabili. Si può accedere a questa funzonalià dal menu *Intellisense*, dal menu contestuale (**Figura 1**) o tramite lo shortcut ctrl-K-X. Molti sono i costrutti che vengono forniti, come vediamo in **Figura 2**.

Fig. 1: **Menu contestuale Expansion**

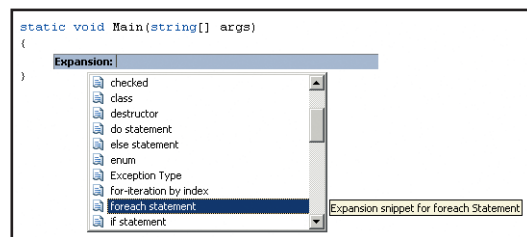



Fig. 2: **Lista di expansions**

In **Figura 3** vediamo le *Expansions* al lavoro: il costrutto viene realizzato, e l'utente può inserire le proprie parti variabili o accettare quelle di default

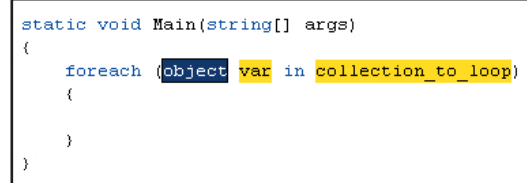



Fig. 3: **Espansione per foreach**

(in giallo). Un comportamento di questo genere era presente in maniera limitata anche in VS2003 per costrutti come i gestori di evento o l'implementazione di interfacce.

```
private int myVar;

public int MyProperty
{
    get { return myVar; }
    set { myVar = value; }
}
```

Fig. 4: **Espansione per property**

Forse la cosa più interessante è che le *Expansions* non sono prefissate, ma sono contenute in file XML contenuti in *C:\Program Files\Microsoft Visual Studio 8\VC#\Expansions\1033\Expansions.* Quindi possono essere modificate dall'utente,che può anche aggiungerne di proprie. Spesso abbiamo dei brani di codice che riutilizziamo frequentemente, e ci piacerebbe avere sottomano. Dal menu *Tools* di VS2005 è possibile accedere allo strumento *Code Snippets Manager* che permette di specificare in quali cartelle risiedano i file XML delle *expansions.*
Alcune espansioni sono particolarmente intelligenti. Per esempio se si aggiunge una property ad una classe come si vede in **Figura 4**, il sistema cambierà in maniera automatica il nome della variabile privata relativa alla proprietà a seconda del nome che le daremo, come vediamo in **Figura 5**

```
private string _indirizzo;

public string Indirizzo
{
    get { return _indirizzo; }
    set { _indirizzo = value; }
}
```

Fig. 5: **Espansione per property modificata**

Surround With *(circonda con)*
Simili concettualmente alle *Expansions*, le funzionalità di *Surround With* permettono di autocomporre codice intorno a codice già esistente. L'esempio più semplice è quello del blocco *try/catch*. Se abbiamo un pezzo di codice che vogliamo mettere in un blocco di controllo per le eccezioni, basta selezionarlo,

```
static void Main(string[] args)
{
    foreach (string s1 in args)
    {
    }
}
```

Refactor ▸
IntelliSense ▸ — Insert Expansion... / Surround With...
Cut
Copy
Paste
Breakpoint ▸

Fig. 6: **Menu contestuale Surround With**

```
System.Console.WriteLine(s1);
Surround With: t
  lock statement
  namespace
  Reverse for-iteration by index
  struct
  Try Catch          Expansion snippet for try catch
  Try Finally
```

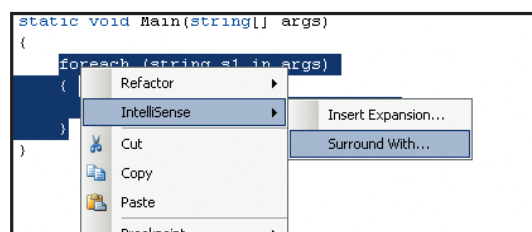
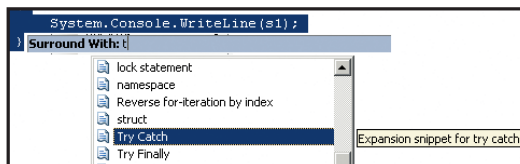



Fig. 7: **Lista Surround With**

selezionare dal menu contestuale Surround With (**Figura 6**) e selezionare dalla lista *try/catch* (**Figura 7**). Il risultato sarà quello in **Figura 8**.

```
try
{
    foreach (string s1 in args)
    {
        System.Console.WriteLine(s1);
    }
}
catch (System.Exception)
{

    throw;
}
```

Fig. 8: **Creazione blocco try**

Lo shortcut per attivare questa funzionalità è *ctrl-K-S.*

Change Tracking *(tracciamento dei cambiamenti)*
Scrivendo codice con VS2005 la prima cosa che salta all'occhio è che il bordo sinistro è colorato, come di vede in **Figura 9**. Questo colore permette allo sviluppatore di sapere quali brani del codice che ha scritto sono stati salvati e quali no. I primi sono bordati in verde, i secondi in giallo. Eventuali pezzi di codice presenti all'apertura del file sono bordati in grigio.

```
static void Main(string[] args)
{
    try
    {
        foreach (string s1 in args)
        {
            System.Console.WriteLine(s1);
        }
        switch (args[0])
        {
            case "1":
                f1();
                break;

            default:
                f2();
        }

    }
    catch (System.Exception X)
```

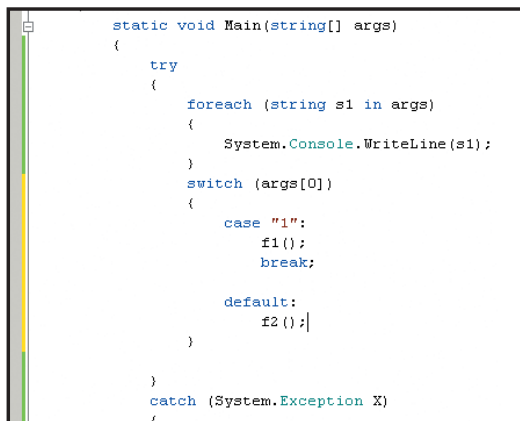
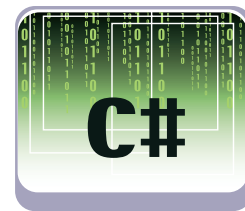



Fig. 9: **Change tracking**

Generate Method Stub (*genera struttura di un metodo*)
A volte è utile scrivere chiamate a metodi non ancora implementati per chiarirci la struttura del codice. Per esempio nel codice di **Figura 9** abbiamo chia-

CI SONO DIFFERENZE NEL COMPILATORE DI QUESTA VERSIONE CON LA VERSIONE STANDARD?
No, il compilatore è esattamente identico a quello contenuto nelle altre versioni di Visual Studio 2005.

SOLO C# SARÀ DISPONIBILE NELLA FAMIGLIA VISUAL EXPRESS?
No, esistono anche Visual Basic 2005 Express Edition, SQL Server 2005 Express Edition, Visual J# Express Edition e Visual Web Developer 2005 Express Edition, ovvero una versione specifica per ASP.NET 2.0.

DOVE POSSO SCARICARE LA BETA DI VISUAL C# 2005 EXPRESS EDITION?
Il sito di riferimento è http://lab.msdn.microsoft.com/express/vcsharp/default.aspx

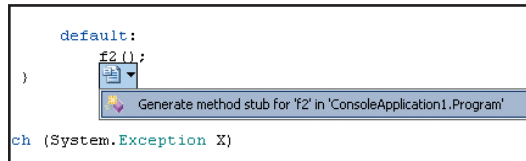
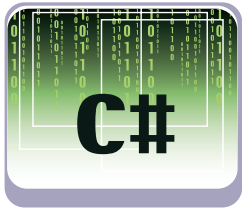



Fig. 10: SmartTag per Genrate Method Stub

mato la funzione *f2()* che però non avevamo scritto. Muovendo il cursore sopra la chiamata si attiva uno smart tag, ovvero un segno sotto la chiamata. Cliccandoci sopra, come in **Figura 10**, ci viene proposto di generare uno stub per il metodo non ancora creato. Accettando, viene automaticamente creato il codice di **Figura 11**.

```
private static void f2()
{
    throw new NotImplementedException();
}
```

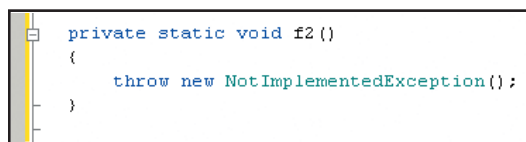



Fig. 11: Codice generato

Intellisense filtrato

Alcune funzionalità dell'Intellisense sono state migliorate aggiungendo dei filtri. Ovvero le proposte fatte dall'intellisense rispecchiano l'ambito in cui si trova il cursore. Per esempio nel caso di un catch l'intellisense mostra solo i tipi eccezione, come mostrato in **Figura 12**.

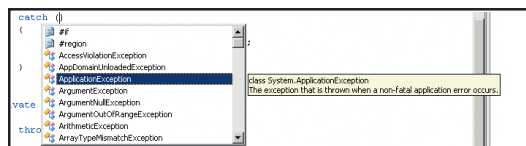



Fig. 12: Intellisense filtrato

Questa funzionalità di filtro si ha anche in altri ambiti, per esempio negli attributi.

Using #region e AutoUsing

In VS2005 le clausole using sono raggruppate in una region. È interessante poi la caratteristica di *"AutoUsing"*, ovvero la proposta che l'IDE ci fa quando andiamo ad usare l'intellisense su una classe per cui non abbiamo richiesto l'uso del corrispondente namespace.

In **Figura 13** vediamo come, avendo scritto *Regex*, l'IDE propone di aggiungere *using System.Text.RegularExpressions;* alla using region o di utilizzare il nome qualificato della classe.

```
private static void f1()
{
    Regex.
}
priv   using System.Text.RegularExpressions;
{      System.Text.RegularExpressions.Regex
                                          on();
}
```

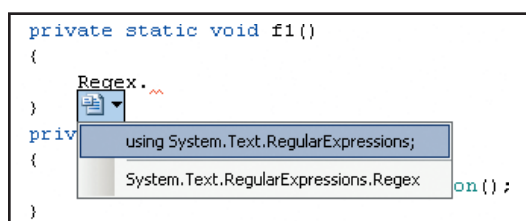



Fig. 13: SmartTag auto-using

Scegliamo la prima opzione, e vediamo il risultato in **Figura 14**.

```
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

#endregion
```

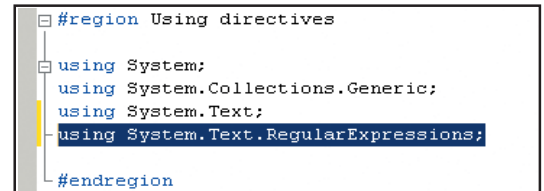



Fig. 14: #region Using

Refactoring

Il *refactoring* è una tecnica che permette, attraverso l'uso di regole, di produrre codice molto chiaro e semplice (si veda *www.refactoring.com*). VS2005 fornisce molte funzioalità di *refactoring*, che verranno più estesamente illustrate in un successivo articolo. Vedremo qui soltanto una delle tante caratteristiche di *refactoring* supportate, quella che ci permette di rinominare in maniera intelligente una variabile. Vediamo un caso concreto.

Abbiamo una variabile che viene utilizzata spesso nel nostro codice, ma che ha un nome che non ci piace. Vogliamo rinominarla, ma utilizzare *Find/Replace* è pericoloso, potremmo danneggiare parti di codice. VS2005 ci permette di fare questa operazione in maniera semplice e sicura. Per esempio vogliamo cambiare il nome della variabile *L1* in **Figura 15**.

```
Libreria libreria = new Libreria();
Libro L1 = new Libro();

L1.Titolo = "invisible";
libreria.QuanteCopie(L1);
if (L1.copie > 0)
{
    Console.WriteLine(L1.Titolo + ": ok");
}
else
{
    Console.WriteLine(L1.Titolo + ": terminato");
}
```

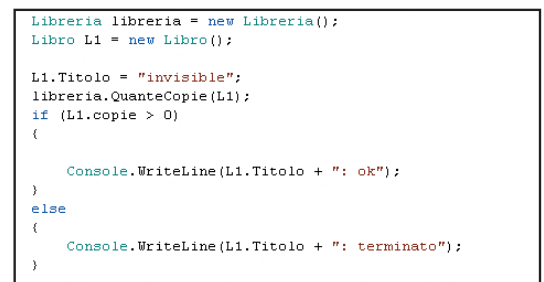



Fig. 15: Codice di esempio

Scegliendo l'opzione Rename del menu contestuale *Refactoring* come in **Figura 16**, si accede alla possibilità di rinominare automaticamente tutte le occorrenze della variabile. In **Figura 17** ci viene chiesto il nuovo nome della variabile, e in **Figura 18** si vede l'anteprima delle sostituzioni che VS intende effettuare.

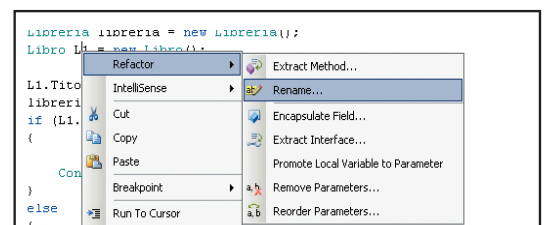



Fig. 16: Menu contestuale rename

SUL WEB

http://lab.msdn.microsoft.com/express/vcsharp/default.aspx
Il sito di Visual C# 2005 Express Edition Beta

http://lab.msdn.microsoft.com/express/vwd/
il sito di Visual Web Developer 2005 Edition Beta

http://lab.msdn.microsoft.com/express/
il sito di Visual Studio Express 2005

www.refactoring.com/
refactoring

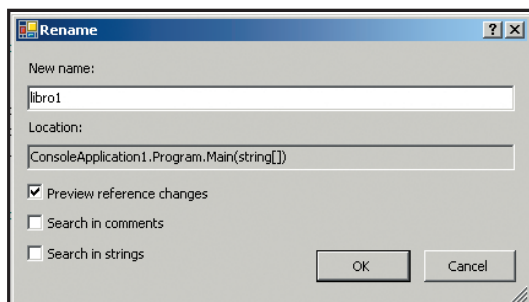



Fig. 17: Dialog di rename

Queste caratteristiche compaiono in modo automatico un po' dappertutto, grazie agli *Smart Tags*.

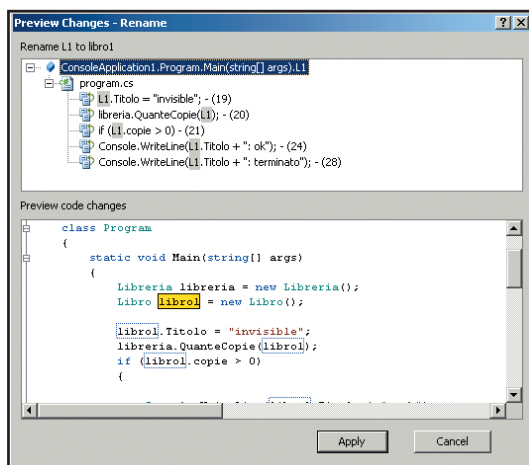



Fig. 18: Anteprima del renaming

Class View

In **Figura 19** vediamo la funzionalità di *Class View*, ovvero la vista della gerarchia di classi che è presente in VSExpress.

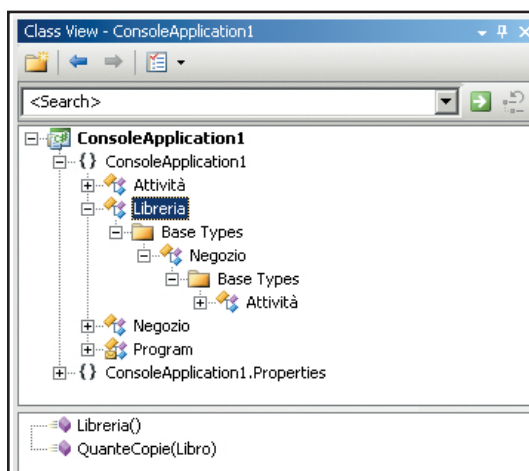



Fig. 19: Class Viewer

STRUMENTI DI INFRASTRUTTURA

Molti sono gli strumenti che compongono l'IDE e che ci aiutano nello sviluppo dei nostri programmi.

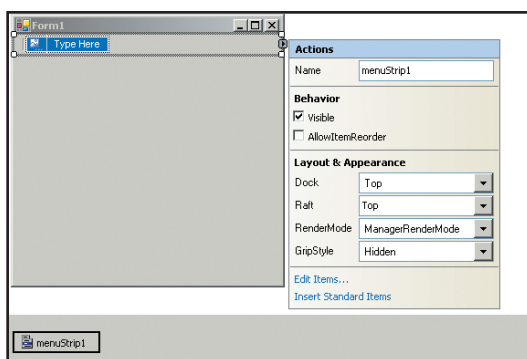



Fig. 20: Menu strip

Prima di tutti, come abbiamo visto, gli *SmartTags*, ma anche i nuovi controlli per WinForms, come il menustrip di **Figura 20**, oppure le opzioni per la formattazione automatica del codice riportate in **Figura 21**.

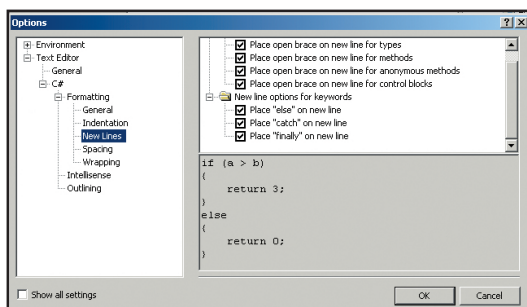



Fig. 21: Opzioni di formattazione

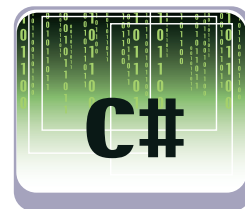
Utile è anche la lista per accedere velocemente ai documenti aperti, riportata in **Figura 22**.

MA ALLORA È VS2005?

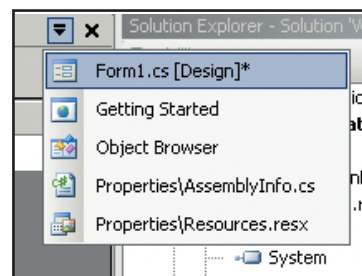
Oltre a fornire aiuti per la stesura del codice, Visual Studio 2005 propone anche strumenti adatti a migliorare la progettazione ad oggetti. Per esempio Il class designer è uno strumento molto avanzato che permette di disegnare le classi dell'applicazione che stiamo costruendo, e di riflettere i cambiamenti sul codice. Il processo è bidirezionale: anche i cambiamenti sul codice si rifletteranno sul diagramma. Purtroppo questa caratteristica di VS2005 non è presente in Visual C# Express 2005, così come il supporto a *Source Safe*, gli *Add-in*, il *ClickOnce Deployment* o lo *Unit Testing*.

Nonostante questo, per coloro che vogliono avvicinarsi allo sviluppo in C# 2, utilizzando uno strumento completo e moderno, Visual C# 2005 Express Edition resta comunque assolutamente da provare.

Marco Poponi,
poponi@innovactive.it

Fig. 21: Menu documenti aperti

L'AUTORE

Marco Poponi è laureato in Ingegneria Elettronica. Insegna Lab di Programmazione alla facoltà di Ingegneria dell'Università degli Studi di Perugia. È tra i fondatori di InnovActive Engineering (http://dotnet.innovactive.it), azienda di sviluppo software e consulenza specializzata nella tecnologia .NET. Si occupa di progettazione, sviluppo e formazione.

Biblioteca

ON LINE

PHP BUILDER

Dedicato agli sviluppatori PHP, questo sito fornisce numerosi articoli e news sempre aggiornate. La sezione più interessante è la code library che raccoglie centinaia di script pronti per essere utilizzati. Non mancate di dare un'occhiata al forum: 60.000 iscritti e mezzo milione di post garantiscono una fonte di informazioni assolutamente impedibile.

www.phpbuilder.com

JAVASTAFF.COM

Portale per la divulgazione e discussione di notizie sul mondo Java. Contiene vari articoli con applicazioni di esempio e permette ai suoi iscritti di pubblicare i propri progetti e collaborare attivamente al sito. Presente anche una sezione download da cui scaricare alcune applicazioni j2me.

www. JavaStaff.com

FLASH GURU

Si può definire spartana la grafica di questo sito, soprattutto se paragonata ad altri siti legati a Flash. Ma la semplicità dell'interfaccia è anche una metafora della chiarezza con cui sono scritti i numerosi tutorial che si possono trovare: i più disparati problemi trovano qui un soluzione spiegata passo passo in modo impeccabile.

www.flashguru.co.uk

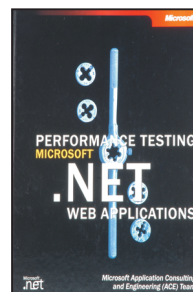
LE RETI - NOZIONI DI BASE

Un testo introduttivo che fa di tutto per "farsi capire". Rivolto principalmente a chi è a completo digiuno dell'argomento, il testo affronta gradualmente tutte le problematiche delle reti di PC: dalla definizione stessa di rete alla implementazione di misure di sicurezza. Attraverso una riuscita similitudine con le reti autostradali, il lettore viene invitato prima a costruire i collegamenti, quindi a seguire il percorso dei veicoli e le ottimizzazioni possibili e così via, fino ad arrivare alle tematiche più complesse e sempre ponendo pari attenzione sia all'hardware sia ai protocolli che caratterizzano le reti. Al lettore non è richiesta alcuna conoscenza specifica, tutti i termini sono ampiamente spiegati con un linguaggio semplice e diretto, nella migliore tradizione divulgativa americana. Ogni capitolo è preceduto da un sommario che elenca le informazioni che saranno illustrate mentre, alla fine dello stesso, è presente un piccolo questionario per verificare il nostro grado di conoscenza. Degno di nota il glossario: ogni voce gode di una descrizione chiara e sintetica al tempo stesso. Indicato per i programmatori curiosi di quello che si nasconde nella Rete… o che vogliano allargare il proprio business!

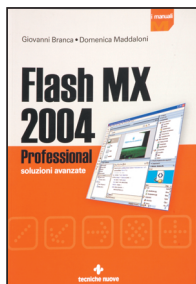
Difficoltà: Bassa • Autore: Wendell Odom • Editore: Mondatori Informatica
http://education.mondadori.it • ISBN: 88-04-53617-9 • Anno di pubblicazione: 2004
Lingua: Italiano • Pagine: 514 Prezzo: € 40,00

PERFORMANCE TESTING MICROSOFT .NET WEB APPLICATIONS

Nel passaggio da applicazioni stand-alone ad applicazioni Web, le prestazioni diventano un fattore critico: servire migliaia di richieste in simultanea richiede conoscenza ed esperienza che non rientrano nel bagaglio di tutti i programmatori. Nel testo sono presentati i principali tool per il test e la profilazione delle applicazioni. IIS, ASP.NET e managed code sono i principali campi di indagine e le metodologie che si acquisiranno sono le medesime che Microsoft utilizza per testare i suoi stessi siti. I principali argomenti:

- Le metodologie di test adottate per Microsoft.com
- Pianificare ed effettuare un test
- Application Center Test di Microsoft per effettuare test di stress
- Monitorare le performance delle applicazioni con il Performance Monitor
- Testare la sicurezza di un sito Web

Al libro è allegato un CD in cui è presente una copia in formato elettronico del libro ed una serie di script di test pronti per l'uso.

Difficoltà: Medio-Alta • Autori: Microsoft Application Consultino and Engineering Team
Editore: Microsoft Press www.microsoft.com/mspress/ • ISBN: 0-7356-1538-1
Anno di pubblicazione: 2004 • Lingua: Inglese • Pagine: 322 • Prezzo: $ 39,99

FLASH MX 2004 SOLUZIONI AVANZATE

Un libro che comincia là dove gli altri si fermano: la programmazione di alto livello in ActionScript. Partendo con un breve capitolo introduttivo sulla programmazione a oggetti in Flash, il libro entra subito nel vivo con un interessante capitolo dedicato ai moduli e alle diapositive, capitolo che si dimostrerà prezioso anche per i programmatori Flash più esperti. Il testo prosegue con grande agilità grazie alle grandi possibilità offerte dai Components: poco codice e molta sostanza, il testo centra in pieno questo obiettivo. Essendo orientata a professionisti, molta cura è dedicata all'interazione con le basi di dati e con XML. Anche grazie agli esempi completi disponibili on-line, il testo riesce a trattare gli argomenti più delicati senza perdere di concretezza ed efficacia. Ciliegina sulla torta, la sezione dedicata allo streaming video: un intero sito dedicato al cinema viene utilizzato come caso di studio.

Difficoltà: Medio-Alta • Autori: Giovanni Branca, Domenica Maddaloni • Editore: Tecniche Nuove
www.tecnichenuove.com • ISBN: 88-481-1518-7 • Anno di pubblicazione: 2004
Lingua: Italiano • Pagine: 244 • Prezzo: € 29,90